AMIGOWIEC

4/1992 Pismo użytkowników komputerów AMIGA cena 10.000 zł

Amiga 600
Pisma, pisemka

Gdzie bije dzwon?
MED cz.2

REALMS!





AMIGOWIEC

kwiecień 1992 cena 10.000 zł.

Redaktor naczelny
Tomasz Kokoszczyński
Z-ca redaktora naczelnego
Ryszard Kowalski
Kolegium redakcyjne
Dariusz Zwierzyński (grafika)
Jacek Ryć (gry)
Tomasz Flanc (Public Domain)
Wojciech Białkowski
Łukasz Bieńkowski
Tomasz Kulbacki
Krzysztof Nowicki
Krzysztof Wirszyłło
Maciej Klimkiewicz
Jarosław Zachwieja
Korektor
Aldona Ossowska

Wydawca

ALFIN, Tomasz Kokoszczyński

Konto: PKO I Bydgoszcz, 9511-64839-136

ADRES REDAKCJI:

AMIGOWIEC
P-17
85-099 BYDGOSZCZ 23

Redakcja nie odpowiada za treść reklam i ogłoszeń.

1 moduł reklamowy kosztuje 90 tysięcy złotych.
Ogłoszenia drobne:
- do 10 słów bezpłatne
- powyżej 10 słów - 2 tys. za słowo

Oto kolejny numer AMIGOWCA. Jaki jest każdy widzi, a że może być lepszy to i my wiemy.

Zapewne ku uciesze wielu z Was udało nam się zwiększyć objętość pisma. Jak to? Że tyle samo stron? No zgoda, ale czcionka jest bardziej ściśnięta i weszło więcej tekstów niż poprzednio. Znowu, niestety, nie udało nam się zrealizować wszystkich zamierzeń na ten numer i wcale nie obiecujemy, że uda się to następnym razem.

Wielu z Was martwi pewnie brak odpowiedzi na listy. Nie będę tu gadał głupot o tym, jak to my dzielnie czytamy Wasze listy i sobie je bierzemy do serca, ponieważ to akurat rozumie się samo przez się. Chcę tylko zaznaczyć, że niektóre odpowiedzi co jakiś czas będziemy zamieszczać i już zamieszczamy (patrz: "Pisma, pisemka"). Na niektóre listy nie jesteśmy w stanie odpowiedzieć, niektóre zaś musimy porozsyłać do naszych specjalistów, co jednak 'trochę' czasu zabiera.

W numerze kwietniowym sporo kontynuacji - jest GdzieBij, jest Kurs C, gra muzyka (MED), są PD-ki. Praktycznie to ten numer się już nam zatkał na wejściu, ale... dzięki 'zwiększeniu' objętości upchnęliśmy trochę więcej. Wielu z Was się głowi: dlaczego te palanty (tzn. my) nie chcą zwiększyć objętości. Uprzejmie odpowiadamy, że nawet gdyby nam się to opłacało (na razie jednak nie chcemy 'ruszać ceny' i to się opłacać nie będzie), to trochę się boimy, że w okresie przejściowym moglibyśmy zaniedbać parę spraw i moglibyście mieć ogromne (uzasadnione z resztą) pretensje. Chwilowo nie widzimy (choć szukamy) rozwiązania tej sytuacji. Gdy coś będziemy wiedzieli na pewno to zauważycie.

Na dziś kończę i tradycyjnie życzę przyjemnej lektury

Tomasz Kokoszczyński.

PS. Dotarł do nas sygnał, że adres wydawcy książki pt. "Amiga DOS" (numer 1/92) został błędnie podany. Oto poprawny adres:
Andrzej Pawłowski
Skrytka Pocztowa 12
00-976 Warszawa

Za błąd serdecznie przepraszamy.

Spis treści

Różne

Gdzie bije dzwon? cz.5

Pisma pisemka

Public Domain

Muzyka

Kurs C

Gry





Nowości Commodora.

Jeszcze nie do końca oswoiliśmy się z najnowszymi produktami Commodore'a jak Amiga 3000 czy CDTV, a już zapowiadany jest następny członek rodziny - Amiga 4000. Według doniesień ma ona być wyposażona w procesor 68040, co pozwoli jej osiągnąć prędkość 5-krotnie większą niż oparta na 68030 Amiga 3000. Poważnym zmianom ulegną również możliwości graficzne i muzyczne tego komputera; ma on standardowo być wyposażony w 24-bitową kartę graficzną (co oznacza paletę ponad 16,7 milionów kolorów) i 16-bitowy układ dźwiękowy (czyli jakość płyty kompaktowej). Prawdopodobnie twardy dysk zostanie podłączony za pośrednictwem interface'u SCSI II, który pozwoli przekazywać dane z prędkością do 10 MB/s (!). Mimo, iż Commodore nie chce podać żadnych szczegółów dotyczących ceny i daty ukazania się maszyny, wygląda na to, iż Apple Macintosh (mimo wprowadzenia nowych modeli Quadra) będzie miał bardzo, bardzo poważnego konkurenta.

Mniej wymagających (lub mniej zamożnych) zwolenników Amigi zainteresują zapewne krążące plotki o modelu Amiga 800. Ma to podobno być zestaw zawierający komputer, twardy dysk i monitor w jednej obudowie (na podobieństwo modelu Macintosh Classic). Mówi się również o tym, że ma on posiadać procesor 68020 i 16-kanałowy generator dźwięku. Ta informacja (w odróżnieniu od poprzedniej) nie jest jednakże pewna...

Sinclair QL.

Czy przypominacie sobie jeszcze dawne dobre czasy, gdy brytyjska firma Sinclair była niemal monopolistą w dziedzinie produkcji komputerów domowych? ZX 81, a później ZX Spectrum na długi czas zdominowały rynek w Wielkiej Brytanii, Niemczech, a nawet w Polsce. W Wielkiej Brytanii komputerek ten ma jeszcze w tej chwili wielu fanów. Potem nadeszła era komputerów 16-bitowych. Sinclair skonstruował wtedy komputer QL. Jego konstrukcja jest oparta na zubożonej wersji mikroprocesora 68000 oznaczonej jako 68008. Od 68000 różnił się on możliwością adresowania tylko 1MB RAM (wtedy było to i tak niewyobrażalnie dużo), miał tylko 4 poziomy przerwań, oraz ośmiobitową szynę danych. Ta redukcja umożliwiła pomieszczenie procesora w 48-nóżkowej obudowie. QL był wyposażony standardowo w 128 KB RAM, świetny BASIC z potwornym edytorem, oraz 1 Microdrive. Studiując Telekomunikację na bydgoskim ATR miałem zajęcia na tym komputerze (w 1988 roku) i z własnego doświadczenia mogę stwierdzić że BASIC bije na głowę najnowsze wersje GFA-Basica (rok produkcji 1991). Dołączony edytor liniowy (porównywalny z EDLIN z PC) jest faktycznie okropny w obsłudze. Jednak ten rewelacyjny na tamte czasy komputer nie odniósł sukcesu. Główną przyczyną był nietypowy nośnik danych Microdrive. Było to skrzyżowanie magnetofonu ze stacją dysków, coś jakby ministreamer. Funkcjonowało to nawet z dość rozsądną szybkością, jednak częste błędy zapisu i odczytu zdyskwalifikowały ten rodzaj nośnika. W pamięci ROM był zainstalowany pakiet programów użytkowych sto razy lepszy niż ten w Commodore Plus 4. Dla tych, którzy chcą przypomnieć sobie tamte czasy, firma UGA napisała emulator QL na Amigę. Ważny jest (na razie tylko dla miłośników legalizmu) fakt, że jest on rozprowadzany jako Public Domain. Emulator działa na wszystkich Amigach (z turbokartą 50 MHz też). Basic na A3000 jest nawet szybszy niż Amiga Basic. Najnowsza wersja nosi oznaczenie 3.1 i można ją nabyć u każdego handlarza PD na Zachodzie.

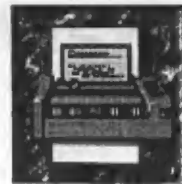
Amiga zamiast odtwarzacza CD ?

Jak wieść niesie, firma Commodore zajęła się poprawieniem niezmienionych od czasu narodzin komputera w 1985 roku możliwości muzycznych. Nowa kość dźwiękowa o nazwie Mary umożliwi obsługę ośmiu kanałów szesnastobitowych sampli stereo. Nie wiadomo jeszcze kiedy Mary zacznie być wbudowywana do seryjnych Amig. Jeśli nie chcesz na to czekać, Drogi Czytelniku, to już teraz możesz zaopatrzyć się w kartę dźwiękową Maestro firmy Macro System, umożliwiającą 16-bitową digitalizację dźwięku. Odtwarzanie odbywa się z rozdzielczością 14-bitową, przez normalne wyjścia Amigi. W porównaniu ze standardowymi parametrami (8 bitów na wejściu i wyjściu) jest to rewelacja. Maestro umożliwia digitalizację dźwięku z płyty kompaktowej bez straty jakości! Dołączony do karty program pozwala na dalszą obróbkę dźwięku (operacje blokowe, filtry, montaż itd.) oraz konwersję na standard ośmiobitowy, przez co samplingi można wykorzystać w normalnych programach muzycznych i na komputerach nie wyposażonych w Maestro. Konwersja psuje oczywiście ich jakość, ale i tak rezultat jest lepszy niż osiągnięty przy pomocy samplera ośmiobitowego. Niektóre firmy software'owe zapowiedziały już wypuszczenie programów wykorzystujących w pełni możliwości Maestro. Karta jest prawdziwym pamięciożercą. Wymaga komputera Amiga 2000 lub 3000 i minimum 3 MB RAM. Koszt karty - ok. 300 DM.

Rośnie biblioteka Soft-Logica.

Oczywiście chodzi tu o bibliotekę czcionek, które mogą być wykorzystane przez jeden z najlepszych programów DTP na Amigę, jakim jest właśnie wyprodukowany przez tę firmę "PageStream 2.x". Biblioteka zawiera 600 pozycji. Fonty, oprócz kilku, są w formacie PostScript typ I. Dzięki PageStreamowi mogą być one 'drukowane' zarówno na drukarkach wysokiej jakości (np. laserowych lub natryskowych) jak i na drukarkach igłowych. Bibliotekę podzielono na 4 grupy:
NewsLetter Fonts (6 czcionek - 100$)
Starter Fonts (6 czcionek - 100$)
Classic Fonts (16 czcionek - 200$)
Designer Fonts (16 czcionek - 200$)

T. Kokoszczynski,
T. Kulbacki i R. Kowalski.



AMIGA 600

Na odbywających się właśnie (piszę to 13 marca) targach CeBIT w Hanowerze, Commodore zademonstrował swój najnowszy produkt **AMIGĘ 600**. W sprzedaży powinna się ona pojawić już pod koniec tego miesiąca, więc zniecierpliwionym potencjalnym nabywcom należy się nieco informacji na temat parametrów technicznych, wad i zalet nowej wersji naszej "przyjaciółki". Już z pierwszym spojrzeniem na komputer uwidaczniają się różnice w stosunku do Amigi 500 i Amigi 500 Plus. Brak klawiatury numerycznej i zmniejszone gabaryty oszczędzają miejsca na stole i sprawiają że nowa Amiga przypomina nieco komputer klasy Laptop. Dokładne oględziny ujawniają dalsze różnice, takie jak wąska szczelina z lewej strony obudowy tam, gdzie przyzwyczajeni jesteśmy widzieć Expansion Port, czy wyjście modulatora TV na tylnej ścianie. Podobnie jak wygląd, zmieniło się również wnętrze komputera. Wszystkie kości montowane są na płycie techniką SMD (Surface Mounted Device), czyli montażu powierzchniowego. Technika ta umożliwiła zaoszczędzenie wiele miejsca na płycie i co za tym idzie zmniejszenie wymiarów komputera. Do przeszłości będą już należeć trapiące użytkowników problemy kontaktu pinów z podstawką. Wady tej nowoczesnej technologii odkryją co bardziej pechowi użytkownicy Amigi (a raczej ich portfele) wtedy, gdy któryś z układów się zepsuje. Każdy, komu popsuł się np. Action Replay zrozumie co mam na myśli. Nawet najczęściej "padające" układy CIA 8520 nie są osadzone na podstawkach. Zastosowanie techniki SMD uniemożliwia naturalnie zastosowanie wewnętrznych rozszerzeń, takich jak ATonce, flicker-fixery czy karty turbo. Po prostu brak podstawki, w którą możnaby je włożyć. Z tego samego powodu nie działają urządzenia przyłączane do Expansion Port, gdyż takowego po prostu nie ma! Gniazdo rozszerzenia pamięci również nie jest kompatybilne ze swoim odpowiednikiem w A500 i A500+. Zamiast Expansion Port, na lewej ściance komputera znajduje się szczelina przewidziana dla Flash Memory Cards. Są to karty wielkości karty kredytowej, zawierające pamięci odpowiednio typu Flash-RAM, Static-RAM lub ROM. W przypadku wykorzystania karty ROM, szczelina ta spełnia rolę gniazda cartridge podobnie jak w C-64, czy Atari XL,XE. Pamięć RAM znajdująca się na karcie nie konfiguruje się automatycznie, ale jej obecność i wielkość jest rozpoznawana przez system. Amiga 600 może bootować zarówno z dyskietki, dysku twardego, jak i z karty ROM. Zamiast typowego gniazda rozszerzenia pamięci jak, na spodzie komputerów A500 i A500+, A600 posiada inne, zmienione i wzbogacone o trzy linie przerwań i jedną Reset. Do ewentualnego zegara zarezerwowano 8 połączeń. Już odmienna forma złącza uniemożliwia podłączenie typowych kart pamięci i Power PC Board.

W przeciwieństwie do złącz, w elektronice nie zmieniło się prawie nic. Większość rozwiązań hardware`owych została żywcem przeniesiona z Amigi 500+, więc kompatybilność z tym modelem została zachowana. Jedynym nowym Custom-Chipem jest Gayle. Zastąpił on układ Gary, a od swego poprzednika różni się możliwością sterowania Flash Memory Card Port i zintegrowanym kontrolerem twardego dysku wg. standardu IDE-Bus. Nowy jest także modulator TV na płycie. A600 posiada standardowo 1 MB CHIP-RAM w postaci wlutowanych w płytę dwóch 4-megabitowych kości HM514260.

Złącza Amigi 600.

1. Złącza zewnętrzne:
   - -Disk Port umożliwiający podłączenie do trzech zewnętrznych stacji dysków;
   - -Centronics;
   - -RS-232;
   - -gniazdo zasilacza;
   - -dwa porty joysticków;
   - -wyjście antenowe wysokiej częstotliwości;
   - -wyjście sygnału FBAS;
   - -Flash Memory Cards Port;
2. Złącza wewnętrzne:
   - -gniazdo rozrzerzeń pamięci;
   - -Disk Interface do wewnętrznej stacji dysków;
   - -IDE-Bus;
   - -Złącze klawiatury.

Wewnątrz komputera znajduje się specjalnie przewidziane miejsce na zamontowanie twardego dysku 2,5" standardu IDE-Bus. Dzięki zastosowaniu układu Gayle odpada konieczność zakupu kontrolera. Oszczędność ta jest jednak z nawiązką wyrównywana przez zabójczo drogie twarde dyski 2,5". Równolegle oferowana wersja **A600 HD** ma taki dysk już wbudowany. Amiga 600 posiada najnowszą wersję systemu operacyjnego Kickstart 2.05 wersja 37.300, co stanowi postęp o 0.125 w porównaniu z Amigą 500+. Ta wersja systemu posiada zintegrowane oprogramowanie do obsługi Flash Memory Cards Port (carddisk.device). Na dysku dołączonym do komputera znajduje się Workbench 2.05.

Amiga 600 będzie kosztować mniej więcej tyle samo, co Amiga 500+, czyli ok. 800-900 DM (w RFN). Commodore zamierza produkować nadal poprzednie wersje Amigi, a o tym, która odniesie sukces, zdecyduje rynek.

Tomasz Kulbacki





GDZIE BIJE DZWON ?

CZĘŚĆ 5

Wprowadzanie danych do pamięci komputera nie jest w dzisiejszych czasach niczym skomplikowanym. Co więcej, pojawiają się coraz doskonalsze i bardziej zaawansowane urządzenia służące do wprowadzania danych, takie jak czytniki dysków optycznych, czytniki różnego rodzaju kodów (kodów paskowych etc.) czy też twarde dyski z wymiennym nośnikiem. Można więc z dużym przybliżeniem założyć, że problem wprowadzania danych do komputera został rozwiązany. Postęp technologiczny w informatyce jest stały. Jednakże wielkie innowacje jakościowe wykorzystywanego sprzętu dokonują się wyłącznie drogą wynalazku. Wiadomo też, na nasze nieszczęście, iż coraz trudniej wymyślić coś rewelacyjnego - poprzeczka genialności wynalazku została podniesiona już bardzo wysoko. Problem wczytania danych pozornie tylko kończy się w momencie zakończenia transmisji z urządzenia wczytującego. Pytaniem jest...

...JAK TO WSZYSTKO UŁOŻYĆ?

W zależności od wymagań pisanego programu musimy starać się tak dobrać sposób składowania danych wykorzystywanych przez program, aby ich obsługa była optymalna. Zazwyczaj dane składuje się tak, aby dostęp do nich był najszybszy. Czasem by ich obsługa była jak najwygodniejsza. Kiedy indziej zależy nam na powiązaniu pewnych danych ze sobą i reprezentowaniu ich w strukturach. Różne potrzeby wiążą się z różnymi rozwiązaniami tego problemu. Poznajmy może ...

...PODSTAWOWE SPOSOBY SKŁADOWANIA DANYCH.

Struktury danych możemy podzielić na **STATYCZNE** i **DYNAMICZNE**. Struktury statyczne mają "na stałe" przypisane w pamięci komputera miejsce, w którym "zapamiętują" reprezentowane dane. Nie można ich więc zmodyfikować w trakcie wykonywania obliczeń - jest niemożliwe dostosowywanie ich do potrzeb programu w trakcie jego realizacji. Sprawą elementarną staje się taki dobór wielkości struktur statycznych, aby wszystkie dane w nich zmieściły się.

Struktury dynamiczne są, jak sama nazwa wskazuje, bardziej elastycznym (choć zazwyczaj trudniejszym) narzędziem w rękach programisty. Specjalny sposób zadeklarowania tych struktur pozwala nam na zmianę ich wielkości - umożliwia to bardzo wygodne projektowanie programu i obsługę danych. Niebezpieczna jest tu obsługa "rozrostu" i "redukcji" struktury danych w miarę potrzeb programu - wszelkie błędy nie pozostaną bez konsekwencji!!!

Należy rozróżnić możliwość modyfikowania zawartości struktury danych oraz modyfikowania samej struktury. Statyczną strukturę danych można porównać do pojemnika na jajka, z którym idziemy na zakupy - ma on ustaloną pojemność, a więc można do niego władować pewną ściśle określoną ilość jaj. Zawartość pojemnika można modyfikować - mogą być tam jajka pierwszego lub drugiego gatunku, może ich tam w ogóle nie być etc. Oczywistą sprawą jest, że pojemnika nie da się "zmodyfikować" i na przykład przystosować do transportowania trzykrotnie większej ilości jaj niż pierwotnie. Strukturę dynamiczną można natomiast porównać do ciągu połączonych ze sobą wagonów kolejowych przeznaczonych do transportowania jakiegoś ściśle określonego typu towarów. Przez przyłączanie nowych wagonów lub odłączanie tych należących do składu można modyfikować strukturę "pociągu danych".

A OTO KONKRETY

Kłócić by się można, czy pojedyncza komórka pamięci (reprezentująca jakąś zmienną) jest już strukturą danych czy nie. Z jednej strony reprezentuje daną, z drugiej jednak trudno mówić tu o jakiejkolwiek strukturze. Pomińmy jednak ten problem milczeniem i przejdźmy zaraz do omawiania bardziej zaawansowanych sposobów reprezentowania danych w pamięci komputera.

Podstawową złożoną strukturą statyczną jest **TABLICA**. Jest to ciąg "miejsc" w pamięci o **Z GÓRY ZADEKLAROWANEJ** długości, w których to miejscach przechowuje się dane o **Z GÓRY OKREŚLONYM TYPIE**. Oczywistym jest, że część tablicy może być pusta, jeśli reprezentuje się w niej mniej danych niż możemy. Tragiczną w skutkach może być próba upchania w tablicy większej ilości danych niż zadeklarowaliśmy w jej długości. Podstawową wadą takiego składowania danych jest mała elastyczność w momencie zmiany wymagań wobec programu. Deklarując tablicę o długości np. 100 elementów, narzucamy programowi pewne ramy, których przekroczenie może się zakończyć zawieszeniem się systemu. A przecież w momencie pisania programu często nie jesteśmy w stanie powiedzieć, czy nie będziemy przypadkiem chcieli zapamiętać w naszej tablicy więcej elementów. Dlatego też tablice do danych, których ilość trudno przewidzieć, deklarować należy z ogromnym zapasem. Jest to niestety bardzo "pamięciożerne" rozwiązanie. Tablice mają także swoje zalety!!! Są nimi: wygodny i szybki dostęp do **KAŻDEGO** elementu zapisanego w strukturze danych, proste indeksowanie tych elementów (numer elementu w tablicy jest zarazem jego numerem kolejnym) oraz bezproblemowe zarządzanie. Warto zadać sobie pytanie co umożliwia tak szybki dostęp do każdego elementu tablicy. Otóż tablica w pamięci komputera reprezentowana jest przez ciąg jej elementów (pewne continuum elementów o określonej długości). Znalezienie n-tego elementu w tablicy jest więc bardzo proste. Wystarczy do adresu początkowego tablicy dodać (n-1) razy długość elementu i już znamy adres n-tego elementu tablicy.

Na bazie tablic często zorganizowane są inne, bardziej wyspecjalizowane struktury danych. Najbardziej znaną spośród nich jest na pewno **STOS**. Jest to struktura, którą z angielska nazywa się **LIFO** (last in first out - czyli kto ostatni wchodzi, ten pierwszy wychodzi). Tu nie można już dobrać się do każdego elementu zapisanego w strukturze. Dostęp



mamy tylko i wyłącznie do zapisanego na końcu stosu (dopisywać i odczytywać można tylko w tym miejscu). Podobieństwa do tablicy są z pozoru niewielkie. Niemniej stos, tak jak i tablica, ma ściśle określoną długość oraz w pamięci jest reprezentowany przez pewien ciąg (continuum) elementów. Podobną, choć nieco inną jest struktura **KOLEJKI - FIFO** (first in first out - kto pierwszy wszedł, pierwszy wyjdzie). Tu mamy możliwość odczytu tylko i wyłącznie z przodu struktury, a zapisu na jej końcu. Trzeba zdać sobie sprawę, że "początek" i "koniec" omawianych powyżej struktur nie oznacza ostatniego elementu stosu czy kolejki, lecz pierwszy wolny element w ich obrębie. Wychodzenie poza ramy stosu i kolejki (przy dobrych układach) kończy się krachem systemu...

Zarówno stos jak i kolejkę można także zdefiniować przy pomocy dynamicznych struktur danych. Aby jednak zrozumieć, iż jest to nieco niewygodne należałoby poznać...

IDEĘ DYNAMICZNEGO ZARZĄDZANIA PAMIĘCIĄ

Dynamiczna struktura danych musi posiadać pewne mechanizmy, umożliwiające jej rozrost (dołączanie wagonów do pociągu) oraz redukcję (ich odłączanie). Dokonuje się tego w dość prosty sposób - każdy element jest złożony z kilku części spełniających różne funkcje. Jedna z nich musi spełniać funkcję wagonu, czyli przechowywać zawartość. Druga musi spełniać funkcję zaczepu, czyli "haka", którym doczepiony jest do kolejnego wagonu. Od biedy może to wystarczyć, choć może się zdarzyć, iż tych składowych będzie dużo więcej. Wbrew pozorom ważniejszą funkcję w dynamicznych strukturach danych spełnia ów hak, który reprezentowany jest przez tzw. wskaźnik (po prostu zmienną, która określa pod jakim adresem w pamięci znajduje się następny element struktury). Powyższy przykład opisuje strukturę tzw. **LISTY JEDNOKIERUNKOWEJ**, w której w każdym elemencie listy zapisana jest informacja o położeniu kolejnego elementu. Nic jednak nie stoi na przeszkodzie, aby w elemencie struktury dołożyć jeszcze jeden wskaźnik, który wskazywał będzie na poprzedni element listy. Zysk z takiego rozwiązania jest ogromny!!! Ponieważ, w przeciwieństwie do tablicy, elementy dynamicznych struktur są rozrzucone po pamięci komputera. Nie możemy więc bezpośrednio obliczyć adresu n-tego elementu struktury. Możliwe jest jedynie "przeskakiwanie z wagonu na wagon" i to tylko w kierunku określonym przez "haki" tzn. wyznaczonym przez wskaźniki. Jest więc sprawą niebagatelną, czy będziemy mogli się poruszać po naszych danych tylko w jedną stronę czy też w obie. **LISTA DWUKIERUNKOWA** daje nam właśnie taką możliwość.

ZYSKI I STRATY Z DYNAMICZNEGO ZARZĄDZANIA

Po stronie plusów stoi ogromna elastyczność tego sposobu zarządzania danymi. Przede wszystkim wcale nie musimy w momencie pisania programu decydować się na jakąś ściśle określoną długość listy. Lista tworzona jest na bieżąco - gdy trzeba dopisać do niej nowy element przydziela się (alokuje) pamięć na kolejny element listy (czyli "buduje się wagon"). Wpisuje się do niego dane ("ładujemy wagon") oraz aktualizuje wskaźniki nadając im takie wartości, aby dopiąć nowy element do całej struktury ("spinamy wagony"). Analogicznie, lecz na odwrót, usuwamy element z listy. Co cenne, ogromnie łatwo jest dopisać element w środek listy (w przypadku tablicy jest to dość kłopotliwe - a dlaczego - to pytanie do Was). Po alokacji i wpisaniu danych możliwe jest nadanie wskaźnikom takich wartości, aby osiągnąć żądany rezultat.

Minus jest chyba tylko jeden - nieuważne projektowanie i programowanie struktury może spowodować błędne działanie programu. Wystarczy na przykład usunąć jakikolwiek element ze środka listy zapominając zaktualizować wskaźniki jego sąsiadów tak, aby omijały likwidowany element. Doprowadzi to do utraty całego "ogona" za usuniętym elementem. Można zwrócić uwagę na jeszcze jedną niedogodność dynamicznych struktur danych. Jest nią niewygoda obsługi, która wynika z bardziej skomplikowanego adresowania elementów struktury. W tablicy każdy element miał z góry znany adres w pamięci - tutaj jest to sprawą niewiadomą. Wszelkie więc algorytmy, np. sortowania, są dużo prostsze do zapisania na strukturach statycznych niż na dynamicznych. Co też zobaczymy już wkrótce...

W następnym odcinku (wbrew pozorom) nie będę pisał o sortowaniu. W jednym lub dwóch odcinkach **"DZWONU"** postaram się przybliżyć Wam zasadę działania systemu operacyjnego. A doprawdy, jest o czym opowiadać..

Marek Hyla

Uwaga! nowy KOKURS

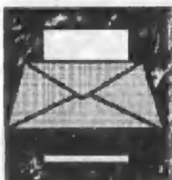
TWIN™ SPARK SOFT.

Jak przed dwoma miesiącami znowu gości u nas firma Twin Spark Soft, która tym razem przygotowała dla Was kolejną porcję piętnastu pytań.

Odpowiedzi prosimy nadsyłać na adres redakcji. Główną nagrodą w konkursie jest jak poprzednio paczka dysków.

Listę zwycięsców poprzedniego konkusu opublikujemy w przeszłym miesiącu.

1. Podaj przynajmniej dwa tytuły komercjalnie rozprowadzanych gier napisanych przez Polaków na Amigę?
2. Na czym w dwóch zdaniach polega zasada działanie standardowej myszy do Amigi?
3. Czy się różni urządzenie o symbolu A590 od urządzenia o symbolu A690?
4. Czy istnieją na Amigę gry umożliwiające połączenie ze sobą dwóch komputerów i zabawę w dwie osoby na dwóch maszynach? Jeśli tak to jakie?
5. Jakie napięcia zasilają płytę Amigi?
6. Ile bajtów zawiera jeden blok na dysku?
7. Podaj przynajmniej 3 tytuły gier firmy CINEMAWARE.
8. Czym zewnętrznie (po załączeniu do sieci) różnią się Amigi wyprodukowane w roku 1987 od produkowanych teraz?
9. Amiga 500 ma procesor Motorola 68000, Amiga 3000 ma procesor Motorola 68030. Jaki procesor posiada Amiga 2000?
10. Czy program DeLuxe Paint umożliwia tworzenie animacji?
11. Ile żył kabla od joysticka jest wykorzystywanych do współpracy z Amigą?
12. Co kryje się pod "skrótami" (może lepiej "napisami") 'df0:' i 'ram:'?
13. W jaki sposób nie dotykając myszy przełączyć aktualnie wyświetlany screen na alter natywny, istniejący pod wyświetlanym?
14. Gdzie na WorkBenchu znajduje się program do formatowania dysku?
15. Do czego służy klawisz z dwiema strzałkami nad klawiszem Ctrl?



Po dwóch "chudych" miesiącach działowi Pisma, pisemka udało się wywalczyć należną mu ilość miejsca na łamach Amigowca. Rozpocznijmy więc od kilku spraw natury ogólnej. W wielu Waszych listach spotykaliśmy się z zarzutem zamieszczania małej liczby ilustracji. Spowodowane jest to niewielką ilością stron naszego pisma. Podczas jego składania musimy często decydować czy zamieścić obrazek, czy też skrócić artykuł. Drugie z tych rozwiązań mogło by stać się przyczyną niejasności w publikowanych tekstach. Z tych samych przyczyn nie jesteśmy w stanie publikować w tej chwili dłuższych opisów programów użytkowych oraz gier. Mamy jednak nadzieję, że pozostaniecie nam wierni do czasu, kiedy objętość Amigowca wzrośnie. Nie chcąc zapeszyć nie ujawnimy kiedy to nastąpi, lecz możecie się spodziewać przyjemnej niespodzianki szybciej niż przypuszczacie. I to by było na tyle jeśli chodzi o nas. Przejdziemy teraz do problemów, z którymi borykają się nasi czytelnicy.

** Co to jest "driver" do drukarki ?*

Swoją odpowiedź na to pytanie chciałbym najpierw uogólnić. Pojęcie "driver" nie jest bowiem związane tylko i wyłącznie z drukarkami, obejmuje ono bowiem również inne urządzenia. Są nimi na przykład:
- karty graficzne,
- karty sieciowe,
- streamery,
- plotery.

Ogólnie można powiedzieć, że jest to pojęcie związane z każdym urządzeniem peryferyjnym, podłączonym do naszej Amigi. W praktyce "driver" jest programem kontrolującym pracę urządzenia peryferyjnego (sterującym nim) oraz tłumaczem danych wysyłanych do niego przez inne programy. Wszystkie programy, które korzystają z jakiegokolwiek urządzenia zewnętrznego wysyłają do niego dane w pewnym standardowym kodzie. Zadaniem "drivera" jest więc zamiana tego kodu na kod zrozumiały dla urządzenia, które obsługuje. Mnogość "driverów" (szczególnie do drukarek) wynika z tego, że producenci nie mogą (a raczej nie chcą) przyjąć jednolitego systemu kodowania informacji dla swoich urządzeń. Również producenci oprogramowania nie zawsze są zadowoleni z możliwości oferowanych im przez standardowe "drivery" i powodowani chęcią polepszenia szbkości współpracy (lub jakiegokolwiek innego parametru) wykonują je sami. Na koniec uwaga, która może być traktowana jako ciekawostka. W komputerze jest tylko jedno urządzenie, które nie posiada własnego "drivera", jest nim mikroprocesor. Wszystko co jest do niego podłączone musi być wyposażone w układ sterujący z tym, że dla takich urządzeń jak pamięć, klawiatura, ekran czy też stacja dysków, jest to realizowane na drodze sprzętowej poprzez stosowanie odpowiednich układów elektronicznych.

** Czy można w rozdzielczości 640*512 lub 640*256 uzyskać maksymalną paletę barw i jaki program to robi?*

Do niedawna odpowiedź na to pytanie byłaby negatywna, teraz jednak uległo to zmianie. Znanej na Zachodzie firmie Newtek udało się uzyskać bez żadnych przeróbek sprzętowych 4096 kolorów w najwyższych rozdzielczościach (również w overscanie). Ten nowy tryb graficzny nazwano "Dynamic Hires" i korzystają z niego niektóre nowe programy graficzne. Przykładem takich programów mogą być np. MACROPAINT oraz DIGI VIEW (ten drugi służy głównie do dygitalizacji obrazów z kamery video lub magnetowidu). Oba mają jednak wspólną wadę. Uzyskanie tak wielkiej ilości kolorów na ekranie wymaga dużej pamięci (minimum 2 Mb). Używanie tego trybu graficznego na zwykłej Amidze (500 lub 2000) nie jest specjalnie wygodne ze względu na ilość danych, które musi przetwarzać. Chcąc korzystać z nowych możliwości musimy się nastawić na kupno szybszego procesora jak również dysku twardego, gdyż obrazek w trybie "Dynamic Hires" potrafi zająć nawet 700 kb.

** Czy istnieje możliwość użycia w Workbenchu 1.3 więcej niż 4 kolorów?*

Oryginalny Workbench 1.3 nie daje możliwości użycia więcej niż 4 kolorów. Nie jest jednak wykluczone, że istnieje przerobiona wersja tego programu, która korzysta z większej ilości kolorów. Korzystając z okazji chciałbym poinformować, że w Workbench w wersji 2.xx ma możliwość ustawienia ilości używanych kolorów od 2 do 16. Zachodnie firmy oferują nawet specjalne 16 kolorowe ikonki do większości popularnych programów. Do tej możliwości należy się jednak odnosić z odrobiną sceptycyzmu, albowiem nasza redakcyjna Amiga 3000 po przełączeniu Workbencha w tryb 16 kolorowy zaczęła pracować zdecydowanie wolniej. Podstawowym zaś zadaniem jakie ma spełniać Workbench jest ułatwianie nam życia, a nie jego utrudnianie.

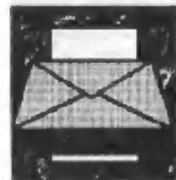
** Czym może być spowodowane przygasanie lub gaśnięcie diody DRIVE w czasie, kiedy stacja wczy-uje lub zapisuje. Czy takie zachowanie się diody jest niebezpieczne dla komputera?*

Nie, takie zachowanie się tej diody nie jest niebezpieczne dla komputera, a spowodowane jest różnicą szybkości z jakimi pracują stacja dysków i komputer. Jeżeli podczas zapisu danych dioda stacji przygaśnie na chwilę oznacza to, że stacja nie otrzymała jeszcze kolejnej porcji danych do zapisu od komputera. Kiedy odczytujemy dane z dysku, przyczyną przygasania diody może być np. fakt wykonywania przez procesor główny naszego komputera innej operacji, stacja musi więc odczekać chwilę aż do momentu w którym będzie on w stanie ją obsłużyć.

** Czasami gdy włączę Amigę oraz włożę jakąś dyskietkę wyskakuje ramka jak w Guru i napis: WARING! AMIGA IS NOT IN A PAL! Co to jest?*

Pojawianie się tego komunikatu jest spowodowane umieszczeniem w bootblocku (fragment dysku, który jest odczytywany i ewentualnie uruchamiany natychmiast po jego umieszczeniu w stacji dysków) dyskietki programiku, który sprawdza w jakim systemie telewizyjnym uruchomił się nasz komputer. Czasami bowiem europejskie wersje Amigi, ni stąd ni zowąd, uruchamiają się w stosowanym w USA i Japoni telewizyjnym systemie NTSC. Komunikat, który się pokazuje informuje właśnie o tym fakcie. Naciśnięcie LMB spowoduje uruchomienie Amigi w trybie NTSC natomiast RMB wykona reset i spróbuje uruchomić system jeszcze raz w trybie PAL.

Należy dodać, że umieszczanie na dyskietkach innych bootblocków niż normalnie jest dość często spotykane i czasami nawet korzystne. Umieszcza się tam bowiem krótkie programiki sprawdzające, czy na naszym dysku nie znajdują się wirusy. Inne typy bootblocków to pro-



gramy ładujące gry, intra (czyli reklamówki ludzi, którzy złamali program lub też go napisali) oraz (co najgorsze) wirusy.

** Czy istnieje możliwość odczytania plików na dyskietce z grą ?*

Taka możliwość istnieje tylko w przypadku, kiedy gra została zapisana w zwykłym formacie stosowanym przez AmigaDOS. W przypadku pozostałych gier (a właściwie ich większości) operacja taka jest niemożliwa. Dyskietka została bowiem nagrana w inny sposób niż normalnie. Wkładając taką dyskietkę do stacji (po resecie komputera) powodujemy uruchomienie się zawartego w bootblocku dyskietki programu, który przejmuje kontrolę lub ignoruje normalne procedury zapisu i odczytu danych z dysku, zawarte w ROM-ie. Jedyną możliwością odczytania zawartości takiego dysku jest rozkodowanie tego programiku, a następnie napisanie na jego podstawie programu narzędziowego, który będzie w stanie odczytać tak zapisaną dyskietkę. Oczywiście nikt tego nie robi, a to z tego prostego powodu, że najczęściej interesują nas nie dane zawarte na dysku (oprócz tego, że są zapisane w specjalny sposób to jeszcze mogą być skompresowane lub zakodowane) lecz to co znajduje się w pamięci komputera po uruchomieniu gry. Mamy wówczas możliwość skorzystania chociażby z Action Replaya.

** Czy istnieją kości ROM z KICKSTARTEM 1.3 lub jego wersja dyskowa dla Amigi 500+?*

Owszem. Istnieje możliwość podłączenia do Amigi 500+ kości z Kickstartem 1.3. W zachodnich czasopismach wiele firm oferuje takie rozszerzenia. Najcześciej spotykanym rozwiązaniem jest płytka drukowana na której umieszczone są układy scalone ze starą wersją systemu oraz miejsce na układy z systemem nowym, które należy przenieść z płyty głównej. Całość umieszcza się w podstawkach kości wyjętych z płyty głównej oraz wyprowadza przełącznik wyboru systemu na zewnątrz. Na drugie pytanie musimy odpowiedzieć negatywnie. Jak dotychczas nie spotkaliśmy się bowiem z dyskową wersją systemu 1.3 dla Amigi 500+. Nie jest jednak wykluczone, że taka wersja istnieje jako, że udało nam się odszukać dyskowy Kickstart 1.3 dla Amigi 3000.

** Czy używanie telewizora jako monitora może spowodować jego uszkodzenie ?*

W praktyce nie ma takiej możliwości. Jedynie uszkodzenie samego komputera lub źle wykonane kable połączeniowe (zwłaszcza podczas korzystania z wejścia EURO telewizora) mogą stać się przyczyną uszkodzenia telewizora.

** Co to jest bufor i do czego służy? Czy jest tym samym co pamięć RAM ?*

Bufor jest pojęciem związanym z większością urządzeń peryferyjnych i oznacza wydzielony fragment pamięci RAM, w którym będą umieszczane dane przesyłane do tego lub z tego urządzenia. Istnienie buforów związane jest z koniecznością zrównania szybkości komputera z np. stacją dysków. Kiedy komputer współpracowałby z nią bez pośrednictwa bufora, musiałby czekać na zapisanie lub odczytanie przez stację danych, a tak może wysłać dane do przydzielonego jej fragmentu pamięci (co trwa o wiele krócej) i zająć się innymi sprawami, podczas gdy przeniesieniem danych do stacji dysków zajmą się natomiast wyspecjalizowane układy Amigi.

** Co to jest streamer?*

Streamer jest urządzeniem służącym do zapisu danych na taśmie magnetycznej (kaseta z taką taśmą jest nieco mniejsza od kasety video). Głównym zastosowaniem tych urządzeń jest wykonywnie kopii bezpieczeństwa dysków twardych o dużych pojemnościach (na taśmie stramera mieści się około 200 Mb). Nie stosuje się go raczej do zapisu danych lub programów do bieżącego użytku z tego samego powodu, dla którego zrezygnowano z używania jako nośnika informacji dla komputerów kaset magnetofonowych, czyli straszliwie długiego czasu dostępu do danych nagranych w ten sposób.

** Co to są samplery i digitalizery ?*

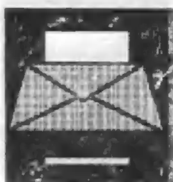
Samplerami i digitalizerami nazywamy urządzenia będące przetwornikami analogowo - cyfrowymi. Digitalizery służą do przekształcania sygałów wizyjnych pochodzących z magnetowidu lub kamery na cyfrowy obraz w wybranej rozdzielczości Amigi, który może być później obrabiany przy pomocy programów graficznych. Samplery zaś działając na podobnej zasadzie przekształcają dzwięk z kompaktu, magnetofonu lub mikrofonu na format cyfrowy, zrozumiały dla programów muzycznych.

** Co to jest MIDI ?*

MIDI (inaczej Musical Instruments Digital Interface) jest to niewielkie pudełko, które podłączone do portu szeregowego Amigi umożliwia jej współpracę z elektronicznymi instrumentami muzycznymi. Instrumenty te muszą być również wyposażone w złącze MIDI. Połączenia dokonuje się z wykorzystaniem pięciożyłowego kabla z wtykiem DIN. Połączenie takie umożliwia korzystanie z zasobów muzycznych Amigi, bez dotykania klawiatury komputera lub też używania Amigi jako urządzenia sterującego tymi instrumentami. MIDI jest niezbędne dla każdego, kto chce w sposób efektywny komponować utwory muzyczne dla naszego komputera. Korzystając z klawiatury instrumentu gramy utwór, który przez złącze MIDI jest przekazywane do Amigi, a następnie może być nagrane na dyskietkę i odtworzone w dowolnym momencie. Oczywiście MIDI to nie tylko elektronika, lecz także programy. W przypadku naszego komputera większość programów muzycznych umożliwia korzystanie z tego złącza (również opisywany w Amigowcu MED 3.0). Jako ciekawostkę można podać, że niektórzy profesjonalni muzycy używają Amigi z opisywanym złączem w swojej pracy np. zespół Pauli Abdul.

A teraz problemy poruszone w kilku innych listach, które otrzymaliśmy.

Pierwszy z nich dotyczy dość intensywnie rozmnażającego się na dyskietkach wirusa SADDAM. Kilku czytelników przysłało nam opisy metod jakimi wytępili oni tego wirusa ze swoich zbiorów. Metody te były do siebie bardzo zbliżone i polegały na niszczeniu pliku Disk-Validator (pod który podszywa się SADDAM) znajdującego się w katalogu L dyskietki, poprzez wczytanie go do dowolnego edytora tekstu oraz wpisanie czegokolwiek do jego kodu, a następnie nagraniu pod tą samą nazwą. Inną metodą było nagranie na SADDAMA Disk-Validatora z dyskietki, która tym wirusem zarażona nie jest. Sprawdziliśmy obie metody i okazały się one skuteczne. Mają one jednak dwie wady. Pierwsza -



dysk z którego przenosimy Disk-Validatora lub na którym znajduje się edytor tekstu, nie może być zarażony wirusem (musimy mieć co do tego pewność). Druga - metody te nie dokonują naprawy szkód, które wirus wyrządził na dyskietce. W redakcji do walki z tym wirusem używamy programu SADDY, który pozbawiony jest tych dwóch wad ponieważ wygrywa walkę z SADDAMEM również w pamięci komputera oraz uwalnia od błędów zniszczone sektory na dyskietce. Nie mniej jednak cieszymy się, że jest ktoś, kto chce się dzielić z nami uwagami na temat swoich doświadczeń. Możemy dzięki temu poznać Wasze problemy jak również sami nauczyć się czegoś nowego.

Waszą ciekawość wzbudziła także informacja dotycząca dysku optycznego do Amigi. Wielu z Was posiada bowiem w domu odtwarzacze płyt CD i jesteście zainteresowani, czy ze względu na podobną zasadę działania nie można by ich wykorzystać jako stacji optycznych do Amigi. Niestety nie jest to możliwe - a to z tego powodu, że kompakt nie jest wyposażony w specjalne wyjście o nazwie SCSI, przez które można go podłączyć do komputera (zakładając, że dysponujemy kontrolerem SCSI). Często natomiast producenci stacji dysków optycznych dają im możliwość odczytywania zwykłych płyt CD (możliwości takiego kompaktu są jednak nieco mniejsze niż urządzeń normalnie do tego celu wykorzystywanych).

Ostanio w zachodnich czasopismach pojawiło się kilka informacji o otwarzaczach płyt CD wyposażonych w złącze SCSI, brakuje nam jednak dokładnych informacji czy mogą one współpracować z Amigą.

List od Pana Tadeusza Witko ze Strzegomia:

"Jestem wiernym czytelnikiem AMIGOWCA, a od kilku miesięcy posiadam też AMIGĘ 500 z rozszerzeniem pamięci do 1 MB. Interesują mnie programy użytkowe, a także chciałbym sam stworzyć coś na moim komputerze, brak mi jednak kontaktu z osobami, które mogłyby mi coś podpowiedzieć, również literatura na ten temat jest dość uboga, dlatego też zachęcony Waszą ofertą zadawania pytań zwracam się o wyjaśnienie kilku trudnych dla mnie do rozwiązania problemów, tj:

1. W jaki sposób połączyć animację graficzną, stworzoną np. programem Deluxe Paint, z muzyką - np. modułami w formacie NoiseTrackera? Czy istnieje jakiś gotowy program łączący muzykę i grafikę (oprócz programu do tworzenia dem, takiego jak RSI czy DIGITAL MARKETING), czy też muszę sam napisać procedurę łączącą oba te elementy i umożliwiającą wystartowanie z mojego dysku stworzonego w ten sposób programu? Jak tego dokonać?

2. Posiadam SOUNDSAMPLER monofoniczny (i 1 MB). Przy pomocy programu AudioMaster udaje mi się wgrać tylko kilka sekund muzyki. Co zrobić aby móc wgrać dłuższe fragmenty, a nawet cały utwór, przy posiadanej przeze mnie pamięci?

3. W jaki sposób można zabezpieczyć swój dysk przed przeglądaniem np. w DiskMasterze? Nie chodzi mi tu o procedurę R-W-E-D, lecz o komunikat błędu w bloku 880 (Error validating disk. Key 880 checksum error.).

4. W jaki sposób zapisać do bootbloku krótki program demonstracyjny czy choćby samą grafikę - rysunek czy też planszę z własnym menu."

Zadał Pan pytania, które najczęściej frapują nowych użytkowników Amigi. Postaram się teraz odpowiedzieć na każde z nich:

ad.1. Jeśli chodzi o problem łączenia animacji z muzyką, to można go rozwiązać w trojaki sposób. Wszystko zależy od umiejętności i oprogramowania jakie posiadamy. Najprostszym, choć najbardziej prymitywnym rozwiązaniem jest napisanie ciągu odpowiednich instrukcji w startup-sequence. Wymaga to jednak posiadania kilku programików do odtwarzania modułów czy projekcji animacji. A oto przykład takiego "startupa", w którym najpierw wszystkie niezbędne dane zostały skopiowane do Ram-dysku, a następnie we właściwej kolejności uruchomione:

```
copy c/copy to ram:
ram:copy mod.muzyka to ram:
ram:copy c/songplayer to ram:
ram:copy animacja.anim to ram:
ram:copy c/delete to ram:
ram:copy c/run to ram:
ram:run ram:songplayer ram:mod.muzyka
delete ram:copy
ram:delete ram:songplayer
ram:delete ram:mod.muzyka
showanim -c1 +5 ram:animacja.anim
```

Sposób ten jest jednak dość pamięcio- i czaso chłonny, gdyż trzeba doświadczalnie ustalić moment odpalenia animacji i muzyki. Zapomnijmy więc o nim i spróbujmy dokonać podobnej operacji przy pomocy specjalnie do tego stworzonych programów. Hmm, tak i tu powstaje pewien drobny problem, ponieważ większość programów tego typu wymaga dużo więcej pamięci niż posiadany przez Pana (i nie tylko) nieszczęsny 1 MB. Należą do nich np. SHOW MAKER czy wspomniany w liście RSI DEMO MAKER (łączenie grafiki z muzyką). Wreszcie trzecim, najlepszym wyjściem z tej sytuacji jest własnoręczne połączenie muzyki i grafiki przy pomocy takiego narzędzia jak język programowania. Może to być dowolny język (Assembler, C, Pascal...), wszystko zależy od naszych umiejętności. Dla początkujących polecałbym AMOS BASIC, który został opracowany właściwie do tworzenia prostych gier i dem. I co najważniejsze, istnieje do niego dość obszerna dokumentacja w języku polskim.

ad. 2. Pytanie dotyczące Audiomastera. Domyślam się, że nie zdążył Pan jeszcze dobrze zapoznać się z tym programem. Ja również posiadam 1MB pamięci i mogę wczytywać kilkudziesięciosekundowe fragmenty muzyki. Wszystko zależy od ustawienia odpowiednich parametrów, takich jak częstotliwość próbkowania (SAMPLE RATE) czy obszaru przydzielonej pamięci (SAMPLE SIZE). Natomiast nie ma mowy o wczytywaniu dłuższych fragmentów lub całych utworów. Jest to oczywiście związane z bolesnym problemem braku pamięci (tzw. kompleks jednego megabajta). Nie należy się jednak od razu załamywać, istnieją bowiem programy do "nagrywania" muzyki bezpośrednio na dyskietki. Mamy wówczas do dyspozycji ok. 880 Kb, co pozwala na zapisanie nawet dwuminutowych fragmentów muzycznych. Dyskietki odtwarzane są w podobny sposób jak płyty (sektor po sektorze). Programy, które umożliwiają taki sposób przenoszenia muzyki na dyskietki to: AL SAMPLER i RECORD MAKER.

ad. 3. Ten osławiony błąd, o którym Pan pisze, możliwy jest do uzyskania wyłącznie z poziomu języka programowania. Jest to już naprawdę "wyższa szkoła jazdy" i dlatego mogę jedynie doradzić kupno kilku książek dotyczących Assemblera.

ad. 4. Ostatnie pytanie dotyczy zapisywania dem lub grafiki do bootbloku. Z tym nie powinno być wielu problemów, gdyż istnieje duża ilość programów wykonujących tę operację. W tej chwili przypominam sobie dwa: BootGirl - do wpisywania grafiki i RSI DEMOMAKER, który ma wbudowaną opcję zapisu do Bootbloku wszystkiego, co zostało nim wykonane.

Krzysztof Nowicki
Tomasz Hanc



PUBLIC DOMAIN DISK #4

Oto ukazał się już 4 z kolei Amigowy Publick Domain Pack. Zanim jednak przedstawię jego zawartość to pozwolę sobie do tradycyjnego przypomnienia charakterystyki wybranych form rozpowszechniania programów.

Public Domain
Programy bez zastrzeżonych praw autorskich. Można z nimi robić teoretycznie wszystko: kopiować, użytkować, adaptować, przerabiać wykorzystywać w innych programach.

Freeware
To programy, które mają zastrzeżone prawa autorskie. Wolno je używać i kopiować (ale tylko w całości). Nie wolno ich sprzedawać, a jedynie kopiować za symboliczną opłatą (za kopiowanie).

Shareware
Czyli programy o ograniczonym prawie do ich używania. Prawo do legalnego posiadania programu uzyskać można dopiero po zarejestrowaniu się jako użytkownik. Następuje to, najczęściej, po zapłaceniu autorowi programu określonej sumy.

Demonstracyjne
Niepełne wersje programu, rozpowszechniane wyłącznie w celach reklamowych. Nie wszystkie z nich można kopiować bez ograniczeń.

AMIGOWY PD #4

1. GO-Moku - jest grą strategiczną rozgrywaną między dwoma osobami, na planszy o rozmiarach 19x19 pól. Zadaniem jest ułożenie linii składającej się z pięciu pionów (w podobny sposób jak w grze "Kółko i krzyżyk"). Zwycięzcą jest ten, kto pierwszy ułoży pięć pionów w rzędzie (pionowo, poziomo lub ukośnie).

Aby rozpocząć nową grę, należy wybrać opcję NEW GAME. Podczas rozgrywania partii możemy "poprosić" komputer o wykonanie za nas ruchu (TAKE GO) lub tylko o zasugerowanie go (SUGGEST GO). Naciśnięcie klawisza F1 spowoduje rozegranie całej partii przez komputer. F10 spowoduje wyświetlenie specjalnych parametrów, które odpowiadają za sposób gry komputera. Zmieniając je możemy osłabić lub wzmocnić naszego elektronicznego przeciwnika.

2. Typist - jest programem stworzonym dla ludzi chcących biegle pisać na maszynie i klawiaturze komputera. Napisany został w formie prostej gry, która uczy właściwego "poruszania" się po klawiaturze i ćwiczy szybkość pisania. Program może zostać załadowany zarówno z CLI jak i z Workbencha. Po uruchomieniu, na ekranie pojawia się okienko podzielone na trzy części. Pierwsza, górna część okna zawiera "wyświetlacz", przycisk START i kontroler szybkości. Kiedy naciśniesz klawisz START po prawej stronie wyświetlacza, pojawią się znaki klawiatury, które przesuwać się będą w lewą stronę. Zasadą gry jest przyciskanie klawiszy (lub ich kombinacji) pojawiających się na wyświetlaczu. Jeżeli nie nadążysz z tą czynnością i znaki przesuną się do lewego krańca wyświetlacza, gra zostanie zakończona. Usłyszysz wówczas sygnał dźwiękowy. Jeżeli pomylisz klawisze również zostanie to zasygnalizowane. Kontroler szybkości pozwala na ustawienie szybkości z jaką wyświetlane będą kolejne znaki (od 1-wolno do 50-bardzo szybko). Dwie strzałki obok kontrolera szybkości służą do zmiany poziomu gry. Opcja SpeedUp pozwala na ustawienie sposobu gry, a dokładniej mówiąc częstotliwości z jaką zachodzą zmiany prędkości wyświetlania kolejnych znaków. Mamy do dyspozycji trzy opcje:

1. NONE -brak zmian szybkości
2. MED -zmiany zachodzą co 30sek
3. FAST -zmiany zachodzą co 15 sek

Środkowa część okna zawiera informacje statystyczne dotyczące szybkości pisania podczas gry (ilość przyciśniętych klawiszy na godzinę, napisanych słów na minutę), a także ilości popełnionych błędów. Dolna część okna zawiera sześć opcji kontrolnych służących do wyszczególnienia części klawiatury, którą pragniemy przećwiczyć. Top Row oznacza górną część, czyli wszystkie litery z górnego rzędu, Middle Row - to środkowy rząd, Bottom - ostatni, dolny rząd klawiszy. Upper Case włącza wielkie litery (uzyskiwane poprzez naciśnięcie SHIFT lub Caps-Locka), Numbers włącza numeryczną część klawiatury (0-9), Symbols uaktywnia wszystkie pozostałe znaki, takie jak: / . , ; @ # $ % ^ itd.

3. Lefty Mouse - jeżeli jesteś leworęczny, to być może ten program pozwoli Ci w pełni wykorzystać Twój komputer. Wynik jego działania jest prosty. Zamienia funkcje klawiszy w myszce, w ten sposób, że ludziom leworęcznym łatwiej jest ją obsłużyć. Program może być uruchomiony zarówno z poziomu Workbencha, jak i CLI. W tym drugim przypadku wystarczy napisać:

LeftyMouse lub *run LeftyMouse*
Aby przywrócić standardowe działanie klawiszy, należy ponownie uruchomić program. Szczerze mówiąc



długo wahałem się z zaprezentowaniem tego programu, gdyż pomimo że stanowi on dość ciekawe rozwiązanie, z moich osobistych doświadczeń i obserwacji wiem, że w przypadku ludzi leworęcznych taka zamiana klawiszy myszy wcale nie jest konieczna. Doszedłem jednak do wniosku, że jeżeli programik ten pomoże choćby jednemu z Was, to i tak wart jest opublikowania.

4. **GPrint 2.03** - jest programem, który umożliwia wydrukowanie dowolnego dwubarwnego rysunku w standardzie IFF na każdej drukarce Epson FX lub kompatybilnej. Zaletą tego programu jest wysoka jakość i szybkość wydruku. Uwaga: Program uruchomić można jedynie z poziomu CLI. Na dyskietce, obok pełnej dokumentacji, znajdują się również dwa rysunki demonstracyjne, które posłużyć mogą do przetestowania programu.
A oto przykład:

GPrint GPrintDemo.IFF h s
GPrint GPrintDemo2.IFF h s

GPrintDemo.IFF jest obrazkiem o rozmiarach 960 na 720 punktów, a GPrintDemo2.IFF - 640 na 400 punktów. Oba wyprodukowane zostały przy pomocy programu Deluxe Paint II. Opcja "h" włącza tryb wysokiej jakości druku, a "s" - tryb specjalnego cieniowania. Wypróbuj program wstawiając opcje "m s" (średnia jakość ze specjalnym cieniowaniem), a także z opcją "u" (ultra). Jeśli chcesz zobaczyć wszystkie możliwe opcje programu, napisz:

GPrint ?

A oto sposób posługiwania się GPrintem:

GPrint filename [LMHU] [DGS] [OnnC] [I] [F] [W] [R]

Gdzie "filename" jest nazwą pliku zawierającego obrazek IFF, który chcesz wydrukować. Następne znaki to opcje:

L: Niska gęstość (80 x 72 ppsi)
M: Średnia gęstość (120 x 72 ppsi)
H: Wysoka gęstość (120 x 108 ppsi)
U: Ultra jakość (240 x 216 ppsi)
D: Tryb Draft (z L, M, H, lub U)
G: Dobra jakość (z L, M, lub H)
S: Specjalne cieniowanie (M lub H)
Onn: Zrównanie nn części cala (np. O5 oznacza 1/2 cala)
C: Centrowanie obrazu
F: Wysunięcie wydrukowanej kartyki papieru
W: Użyj jeśli posiadasz drukarkę 13.6"
R: Wyłączenie reversu (dla drukarki MX lub RX)
B: Pełne zaczernienie (tylko z S)

Przykład:
GPrint df1:Picture h s f o15

5. **Directory Opus 2** - na pewno każdy z Was zna program DiskMaster. DirectoryOpus spełnia podobne zadanie, z tą tylko drobną różnicą, że program ten jest o wiele lepszy! Posiada wiele ciekawych, ułatwiających pracę funkcji, spośród których wyróżnić można:
- dwa duże okna widoczne jednocześnie na ekranie monitora,
- dwadzieścia buforów, w których zapamiętywane są katalogi dyskietek (co znacznie wpływa na oszczędność czasu),
- nieograniczoną liczbę edytowanych plików (nie licząc ograniczenia pamięcią),
- czternaście standardowych gadżetów i osiem menu,
- funkcje ustalane w dowolny sposób przez użytkownika,
- możliwość ustawienia własnej konfiguracji programu (palety kolorów, kroju pisma, itd.),
- ponad trzydzieści wbudowanych komend, takich jak: Copy, Move, Rename, Delete, Show, Play, Read, Relabel, Comment, DateStamp, Assign, Print, Format, DiskCopy, Install, Addbuffers, Search, Hunt File i wiele innych,
- możliwość zamiany gadżetów miejscami, co pozwala na ustawienie programu w sposób najbardziej optymalny dla danego użytkownika,
- pełny i łatwy w dostępie inerface ARexx-a,
- Directory Tree - czyli tzw. drzewo,
- Help - czyli bardzo pomocna opcja zwłaszcza dla początkujących,
- możliwość czytania tekstów, wyświetlania obrazków i odgrywania sampli.

__Uwaga:__

Zamieszczony przeze mnie DirectoryOpus 2 to zaledwie demonstracyjna wersja oryginalnego programu komercyjnego. Jak mi wiadomo, powstał niedawno DirectoryOpus 4, w którym znacznie rozbudowano opcje i całkowicie zmieniono panel konfiguracyjny. Prezentowana na naszym dysku wersja to zaledwie przedsmak prawdziwego Opusa. Jego najnowszą, pełną wersję można nabyć w firmie uprawnionej do jego dystrybucji. DirectoryOpus kosztuje około 50 dolarów australijskich i należy zamawiać go pod adresem:

Left Side Software
P.O. Box 289
Goodwood, SA 5034
Australia

ph : (08) 2932788

6. **WiLdDogLand** - to moduł muzyczny napisany przy pomocy programu z rodziny "Trackerów". Został on skomponowany i udostępniony naszej redakcji przez jednego z czytelników - Tomasza Rożka.

7. **JęzykC** - to oczywiście zestaw listingów programów z bieżącego odcinka kursu języka "C" przygotowanego przez Bohdana R. Raua.

Autorem większości ikonek zamieszczonych na dysku jest Dariusz Zwierzyński.

Do wszystkich programów załączone są krótkie opisy w języku polskim !

Na dyskietce znajduje się również spis wszystkich dotychczas opublikowanych programów (plik 'Przeczytaj').

Prosimy o uwagi o naszej bibliotece. Piszcie do nas jakich programów poszukujecie i jakie powinny się w niej znaleźć. Czekamy również na owoce Waszej pracy.

Jak zamawiać i ile to kosztuje ?

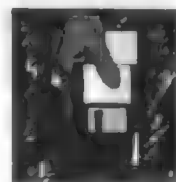
Dyskietki PD można zamawiać przesyłając pieniądze na nasze konto (blankiet na stronie 13) oraz:

- podając numer dysku PD (#1 - #4)
- podając CZYTELNIE swój adres
- podając ilość posiadanej pamięci w komputerze

Nasz dysk PD kosztuje 25 tysięcy złotych (włączone w to są koszty dysku, przesyłki, nalepki, opakowania, itp.).

UWAGA: Dla osób prenumerujących AMIGOWCA cena jest niższa i wynosi tylko 20 tysięcy złotych ! Jeżeli ktoś życzy sobie by mu wysłać dysk PD za zaliczeniem pocztowym, to sam pokrywa koszty tego zaliczenia.

Tomasz Hanc



MED Music Editor v. 3.00

OPIS PROGRAMU CZĘŚĆ 2

Synthetic sound editor

Jest to edytor, z pomocą którego można tworzyć proste fale dźwiękowe oraz "programować" czas i szybkość narastania i opadania dźwięku. Dźwięki syntetyczne (synthetic sounds) zajmują niewiele miejsca w pamięci komputera (to jest ich główną zaletą). Poza tym są bardzo proste i przypominają swoim brzmieniem starego dobrego C-64.

Uaktywnienie

Aby uaktywnić edytor dźwięków syntetycznych należy:

1. Wczytać MED
2. Wybrać panel INSTR
3. Wcisnąć opcję SYNTH
4. Wcisnąć opcję SYNTH-DISPLAY

Po wyłączeniu opcji edycji (EDIT OFF) [redacted] przy pomocy klawiatury odtworzyć konstruowany dźwięk.

Edycja [redacted] dźwiękowej

Na ekranie przedstawione [redacted] dwie fale. Pierwsza (z lewej strony) to "fala główna", druga to "fala pomocnicza" (znajdująca się w buforze). Poniżej znajduje [redacted] szereg opcji pomocnych podczas edycji fali. I tak dwie opcje COPY (lewa i prawa) służą do kopiowania fal. Jeśli naciśniesz lewą opcję COPY, fala znajdująca się po prawej stronie ekranu zostanie skopiowana [redacted] stronę lewą i odwrotnie.

Opcje:

[<- EXCHG ->] - zamienia fale dźwiękowe miejscami

[<- MIX] -miksuje (miesza) obie fale

[<-ADD] - fale łączone [redacted] jedna [redacted] drugą

[<- EDIT] [EDIT ->]- wskazanie edytowanej fali

[UNDO] - cofnięcie ostatniej operacji

[RANGE ALL] -zaznaczenie całej fali

Rysowanie fali

Bardzo wygodną formą kreowania nowej fali dźwiękowej jest rysowanie jej kształtu przy pomocy myszy. Wystarczy tylko wcisnąć lewy przycisk myszy i rysować. Możliwe [redacted] do uzyskania dwa tryby rysowania: punktowy (pixel) i liniowy (line).

Długość fali

Obie fale dźwiękowe mogą zajmować od 2 do 128 bajtów pamięci. Do zmiany długości fali służy opcja "LEN:".

Fale złożone

Instrumenty syntetyczne mogą zawierać wiecej niż jedną falę dźwiękową (maksymalnie 64). Po lewej stronie ekranu, pod graficznym odzwierciedleniem fali dźwiękowej, znajduje się małe okienko zawierające kilka informacji:

```
nr fali bieżącej
       |
       0 < > / 00
         |/    |
  nast./poprz.  nr ostatniej fali
```

Znaki "<", ">" pozwalają przełączać fale dźwiękowe. Tą samą operację umożliwia również odpowiednia kombinacja klawiszy: Alt-strzałka lewo = poprz. Alt-strzałka prawo = nast.

Zaznaczanie obszaru roboczego

Istnieje kilka operacji edycyjnych, które wymagają określenia obszaru roboczego. Za pomocą opcji RANGE ALL można zaznaczyć cały obszar fali. Natomiast przy pomocy prawego klawisza myszy [redacted] się tylko wybrany fragment (RANGE).

Jeżeli nie zaznaczymy żadnego obszaru roboczego, [redacted] ekranie znajdować się będzie biała, pionowa linia. Jest to kursor potrzebny pod[redacted] operacji edycyjnych.

Zmiany pozycji kursora dokonujemy przy pomocy prawego klawisza myszy lub specjalnych opcji:

```
kursor [redacted]          kursor [redacted] koniec
początek |                  |
              <   +   >
                  |
        kursor po środku fali
```

Na środku ekranu znajduje się słowo "RNG", [redacted] obok niego kilka opcji:

RNG Cut Copy Paste Clr Dbl Rev < >

Cut - działa tylko z falą po lewej stronie; wycina zaznaczony obszar, który jest przenoszony [redacted] prawą stronę

Copy - kopiuje zaznaczony obszar na prawą stronę

Paste - kopiuje zawartość prawej fali do pozycji wskazanej przez kursor

Clr - czyści obszar

Dbl - podwyższa obszar o oktawę

Rev - odwraca postać fali zaznaczonego obszaru

< or > - przesuwa zaznaczony obszar

Hybrid sounds

Instrumenty hybrydowe to coś więcej niż instrumenty syntetyczne. Ich podstawą nie jest prosta fa[redacted] dźwiękowa, ale sample, czyli postać cyfrowa jakiegoś dźwięku naturalnego. Do kształtowania instrumentów hybrydowych wykorzystuje się opisany już wcześniej język programowania dźwięku.

Sample editor

Podobnie jak nowsze wersje "trackerów" również MED posiada wbudowany moduł wczytywania i obróbki sampli, czyli tzw. sample editor. Uaktywnienie modułu następuje po wybraniu opcji SMPED. W centralnej części ekranu sample edito[redacted] znajduje się okno edycyjne, w którym wyświetlone zostaje graficzne odzwierciedlenie dźwięku (tzw. waveform). Po wczytaniu do pamięci komputera dowolnego instrumentu, na ekranie edycyjnym sample editora ukaże się wykres charakteryzujący dany dźwięk. W prawym górnym rogu ekranu ilustrującego przebieg [redacted] dźwiękowej znajduje się liczba, informująca o długości fragmentu fali, którą obserwujemy. Podczas, gdy [redacted] ekranie edycyjnym wyświetlany jest cały sample, wartość [redacted] stanowi jednocześnie jego długość. Minimalnie może [redacted] wynosić 312 bajtów. Poniżej ekranu edycyjnego znajduje się czerwona "listwa", która informuje o pozycji i wielkości wyświetlanego fragmentu sampla.

Rozmiar bufora

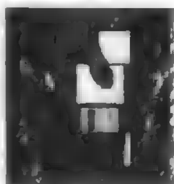
Po lewej stronie ekranu znajduje się opcja BUFF:. Ilustruje ona rozmiar bieżącego sampla. Jeśli chcesz zmienić rozmiar, możesz wpi[redacted] nową wartość. Maksymalny rozmiar bufora wynosi 131072 bajtów.

[redacted] obszaru

Wiele opcji edycyjnych wy[redacted] wcześniejszego zaznaczenia obszaru roboczego, [redacted] którym będą wykonywane jakieś operacje. Zaznaczanie obszaru odbywa się przy pomocy prawego klawisza myszy. Trzymając go zaznaczamy obszar roboczy. Opcje RNGS/RNGE pokazują aktualną pozycję początku i końca zaznaczonego obszaru. Istnieje możliwość zmiany tych pozycji. W tym celu należy przy użyciu klawiatury wprowadzić [redacted] wartości liczbowe. Opcja RANGE ALL pozwala na zaznaczenie całego obszaru wyświetlanego na ekranie.

[redacted]

Następnymi opcjami po BUFF [redacted] PLAY DISP i PLAY RNG. Pierwsza z nich odtwarza



fragment sampla znajdujący się aktualnie na ekranie, a druga odtwarza zaznaczony fragment sampla. Cały instrument (bez względu [redacted] to jaki obszar wyświetlany jest na ekranie) może być odegrany przy pomocy klawiatury.

Powiększanie

Bard[redacted] pomocną opcją, zwłaszcza przy precyzyjnej obróbce dźwięku, jest zastosowanie powiększenia. Istnieje [redacted] możliwości powiększania naszych instrumentów (sampli). Jedną [redacted] nich jest proste włączenie powiększenia - ZOOM IN lub jego wyłączenie - ZOOM OUT. Aby uzyskać powiększenie [redacted]nego obszaru należy wcisnąć SHOW RNG. Zaznaczony fragment zostanie wyświetlony na ekranie. Przywrócenie obrazu całego sampla następuje po wybraniu opcji SHOW ALL. Zresztą opisywane powyżej opcje powinny być dobrze [redacted] użytkownikom AudioMastera. Jeśli powiększenie jest włączone (ZOOM IN) istnieje możliwość przewijania sampla, poruszania się po nim. Umożliwiają to klawisze zmiany kierunków (strzałki: lewo/prawo).

Sampling MED, podobnie jak niektóre "trackery", posiada wbudowany moduł obróbki dźwięku. Funkcja MONITOR ukazuje w czasie rzeczywistym przebieg dźwięku z samplera. Opcja DIGITIZE umożliwia wczytywanie do pamięci komputera dźwięku z samplera. Wciśnięcie lewego przycisku myszy umożliwia rozpoczęcie samplingu. Jego zatrzymanie następuje automatycznie po zapełnieniu się bufora. Można również przerwać ten proces w sposób sztuczny. Do tego przeznaczony jest prawy przycisk myszy.

Sampling/playing rate

Po prawej stronie ekranu znajduje się wartość liczbowa informująca w sposób umowny [redacted] aktualnej częstotliwości próbkowania (standardowo 428). Jest to jednocześnie wartość określająca wysokość wyświetlanego tonu. Można ją zmienić, wprowadzając nową liczbę w miejscu oznaczonym przez PER i naciskając RETURN. Np. aby podwoić szybkość samplingu, należy wpisać liczbę 214.

A oto wykaz wszystkich częstotliwości:

C-1	856	C-2	[redacted]	C-3	214
C#1	808	C#2	404	C#3	202
D-1	762	D-2	381	D-3	190
D#1	720	D#2	360	D#3	180
E-1	678	E-2	339	E-3	170
F-1	640	F-2	[redacted]	F-3	160
F#1	604	F#2	302	F#3	151
G-1	570	G-2	285	G-3	143
G#1	[redacted]	G#2	269	G#3	135
A-1	508	A-2	254	A-3	127
A#1	480	A#2	240	A#3	120
H-1	453	H-2	[redacted]	H-3	113

Zmiana wysokości brzmienia. Jeśli pragniemy transponować instrument o częstotliwości 428 (C-2) w [redacted] sposób, aby dźwięk G-2 znajdował się w miejscu C-2, to musimy wykonać trzy kroki:

1. Upewnić się, że bieżąca częstotliwość wynosi 428.
2. Wpisać wartość 285 (G-2) do PER, NIE NACISKAĆ RETURN !!!
3. Wybrać opcję CHANGE RATE.

W dość prosty sposób można również dokonać transponowania dźwięku o całą oktawę. Operację tę umożliwiają dwie opcje. OCT (strzałka w górę) podnosi sampla o jedną oktawę. Oct (strzałka w dół) transponuje sampla o oktawę w dół.

Wybór formatu zapisu

W dolnym prawym rogu ekranu znajdują się opcje służące do ustawienia formatu zapisu sampli (RAW i IFF). Jeśli opcja IFF jest załączona, instrumenty nagrywane będą w formacie IFF 8SVX. Ten z pozoru skomplikowany zapis oznacza, że informacje dotyczące zapętlenia dźwięku będą zapisane [redacted] z nim.

Podstawowe operacje edycyjne

Sample editor MEDa 3.00 zawiera podstawowe operacje edycji, które występują w większości edytorów sampli. Zlokalizowane są one w dolnym lewym rogu ekranu:

CLR - czyszczenie obszaru bieżącego
DEL- wymazywanie zaznaczonego obszaru (bez skopiowania do bufora)
CUT - wymazywanie zaznaczonego obszaru (z uprzednim skopiowaniem do bufora)
COPY - skopiowanie zaznaczonego obszaru do bufora
PASTE - wyrzucenie obszaru z bufo[redacted] na ekran (w miejscu wskazanym prze[redacted] kursor)
Wyczyszczenie zawartości bufora umożliwia opcja DEL CBUFF.

Operacje dodatkowe to:

1. S->CB kopiowanie całego sampla do bufora
2. CB->S kopiowanie bufora do sampla

Operacje dodatkowe na buforze są bardzo pomocne podczas eksperymentów, do których zresztą gorąco namawiam. Rozliczne próby ulepszania dźwięku mogą prowadzić np. do zagadnień związanych z pogłosem (echem). Poprzez S->CB można skopiować dany instrument do bufora, następnie wykonać kilka operacji związanych z wprowadzeniem echa i wysłuchać osiągnięty rezultat. Jeśli efekt naszych zabiegów wydaje się niezadowalający, można szybko przywrócić dawne brzmienie przez CB->S.

Odwracanie

REV - odwracanie [redacted]nego obszaru

Miksowanie

Miksowanie dwóch dźwięków. Dźwięk źródłowy musi znajdować się w buforze. Teraz należy zaznaczyć pozycję od której ma się zaczynać drugi dźwięk (prawy przycisk myszy). Następnie wcisnąć MIX. Można również zaznaczyć obszar, w którym [redacted] nastąpić miksowanie (trzymając cały czas prawy przycisk myszy).

Zmiana głośności

Efekt ten pozwala na zmianę głośności sampla. Zabiegu tego dokonuje sie przy użyciu opcji CHANGE VOL. Należy jednak zmienić dwie wartości. Lewa to początkowa głośność, prawa - końcowa. Obie stanowią o wartości procentowej oryginalnego dźwięku. Na początku należy zaznaczyć obszar do obróbki. Następnie wystarczy w odpowiedni sposób ustawić wartości parametrów START VOL i END VOL.

A oto przykłady:

1. Podwojenie głośności: ustaw obie wartości na 200
2. Ściszenie dźwięku: ustaw początkową wartość na 100, [redacted] końcową [redacted] 1
3. Dźwięk o wzrastającej głośności (początkowa głośność stanowi połowę końcowej)

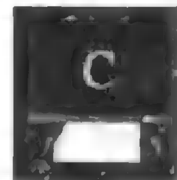
start vol = 50, end vol = 100

Po wprowadzeniu wartości początkowej i końcowej, wystarczy nacisnąć Change Vol i całą brudną robotę wykona [redacted] Ciebie komputer. Jeżeli jednak przesadzisz z wielkością zmian, to dźwięk ulegnie zniekształceniu.

Echo

Dzięki zastosowaniu pogłosu, brzmienie Twoich instrumentów może zostać w znacznym stopniu przekształcone. Echo jest kontrolowane zależnie od dwóch wartości: czasu oraz liczby pogłosów. Czas jest to dystans pomiędzy dwoma pogłosami. Jeśli jest on krótki można uzyskać ciekawe efekty. Druga wartość charakteryzująca echo jest niczym innym jak całkowitą liczbą pogłosów.

Tomasz Flanc



XIV: SZEF KUCHNI PROPONUJE, CZYLI MENU

No, to teraz możemy szaleć. Prawie jak w BASICu, możemy sobie napisać różne ciekawe funkcje czytania klawiatury czy myszy. Do pełni szczęścia (czytaj: profesjonalizmu) brakuje [redacted] tylko jednego drobiażdżku: menu.

Ale najpierw kwestia terminologii. Ponieważ nie mam zamiaru spolszczać na siłę terminów używanych przez twórców systemu *intuition*, na rysunku podaję co jak się nazywa. Myślę, że będzie to prostsze niż pisanie np.: "MenuStrip to linia znajdująca się [redacted] górze ekranu i zawierająca wszystkie nazwy menu".

Rzecz najprostsza - deklaracja menu.

Jeśli spojrzymy do pliku "include/intuition/intuition.h" (a [redacted] szczerego serca, radzę wszystkim zaglądać tam w miarę często), zobaczymy deklarację struktury **Menu**:

```
struct Menu
    {
    struct Menu *NextMenu;
    SHORT     LeftEdge, TopEdge,
              Width, Height;
    USHORT Flags;
    BYTE *MenuName;
    struct MenuItem *FirstItem;
    SHORT     JazzX, JazzY,
              BeatX, BeatY;
    };
```

Co znaczą poszczególne nazwy?

NextMenu: wskaźnik do następnej struktury **Menu** w liście MenuStrip. W ostatniej strukturze wskaźnik ten musi mieć wartość 0;

LeftEdge: Pozycja menu od lewej strony ekranu;

TopEdge: Pozycja menu od góry ekranu (patrz dalej);

Width: Szerokość menu;

Height: Wysokość menu;

UWAGA!

W znanych mi systemach, pola Height i TopEdge nie są używa[redacted]. Menu ma zawsze taką samą wysokość jak linia tytułowa ekranu i zaczyna się od samej góry.

Flags: Przez *intuition* określone [redacted] tylko dwa znaczniki. Znacznik MENUENABLED oznacza włączenie menu (tzn. użytkownik może wybierać pozycje). Zmiany stanu znacznika MENUENABLED powinny odbywać się przez wywołanie funkcji **OnMenu** lub **OffMenu**. Znacznik ten może być bezpośrednio ustawiony przez użytkownika przed wywołaniem funkcji [redacted]enuStrip. Drugi znacznik MIDRAWN ustawiany jest przez *intuition* i informuje użytkownika, że menu w danej chwili jest wyświetlane na ekranie. Pamiętajmy o tym, że Amiga to nie durny IBM i jeśli *intuition* wyświetla sobie menu, nie znaczy to, że nasz program na coś czeka. Może sobie liczyć dalej pod warunkiem, że nie będzie chciał na naszym ekranie nic wyświetlać.

MenuName: wskaźnik do nazwy menu.

FirstItem: wskaźnik do pierwszej struktury MenuItem.

Następne cztery komponenty, [redacted] nazwach wiele mówiących o ubocznych zainteresowaniach twórców *intuition* i, są używane tylko przez system i na początku dobrze jest ustawić je na 0.

Oto przykład inicjalizacji struktury **Menu**:

```
struct Menu ProjectMenu = {
        &EditMenu,    /* wskaznik do
                         nastepnej struktury */
        10,0,      /* polozenie menu [redacted]
                         ekranie */
        60,0,     /* wymiary menu */
        MENUENABLED,  /* znacznik
                         wlaczenia menu */
        "Project", /* [redacted] menu */
        &OpenItem,    /* wskaznik do
                         pierwszego elementu */
        0,0,0,0);
```

Jeśli więc będziemy chcieli napisać np. edytor tekstu, deklaracja menu może wyglądać [redacted] przykład tak jak **listing #1**.

Jak widać, deklarowanie menu jest naprawdę dziecinnie proste. Gorzej natomiast przedstawia się sprawa jego elementów, czyli struktury **MenuItem**.

Sama struktura wygląda następująco:

```
struct MenuItem
{
        struct MenuItem *NextItem;
        SHORT LeftEdge, TopEdge,
              Width, Height;
        USHORT Flags;
        LONG MutualExclude;
        APTR ItemFill;
        APTR SelectFill;
        BYTE Command;
        struct MenuItem *SubItem;
        USHORT NextSelect;
};
```

I oczywiście znaczenie poszczególnych nazw:

NextItem: wskaźnik do następnej struktury **MenuItem** [redacted] liści[redacted]. Nie muszę chyba przypominać, że w ostatniej powinno być 0.

LeftEdge i TopEdge: położenie lewego górneg[redacted] rogu pola selekcji (tego kawałka menu, w którym jak wsuniemy mysz to się nam zaświeci **MenuItem**). Położenie podawane jest względem lewego górnego rogu pola menu (jeśli podamy 0,0 to będzie to akurat w [redacted] dla pierwszego elementu).

Width i Height: wymiary pola selekcji.

Flags: o tym można napisać osobną książkę, [redacted] więc cierpliwpści.

MutualExclude: informuje *intuition* o tym, że przy selekcji danego elementu musi nastąpić automatyczna deselekcja innych. Wartość wstawioną w to miejsce oblicza się w sposób następujący:

Każdemu elementowi (MenuItem) [redacted] liście przyporządkowany jest kolejny numer począwszy od zera. Załóżmy, że w chwili selekcji danego elementu musi nastąpić deselekcja elementów o numerze 0,7 i 11. Obliczamy:

(1<<0) | (1<<7) | (1<<11)

i tę wartość wstawiamy do pola MutualExclude. W naszym przypadku będzie to wartość 0x881.

ItemFill: wskaźnik do struktury **Image** lub **IntuiText** która będzie wyświetlana w miejscu danego elementu.

SelectFill: wskaźnik do struktury, która będzie wyświetlana gdy wskaźnik myszy znajdzie się w polu selekcji danego elementu.

Command: Jeśli mamy zamiar uruchomić dany element [redacted] klawiatury, wpisujemy tu kod znaku. Naciśnięcie tego znaku razem z prawym klawiszem AMIGA spowoduje wysłanie przez *intuition* informacji o selekcji danego elementu.

SubItem: Jak wiadomo, każdy element [redacted] liście menu może rozwinąć swoje własne sub-menu. W tym miejscu podajemy wskaźnik do pierwszego elementu listy **MenuItem**.

LISTING #1

```
struct Menu SkokMenu = {
        0,21,0,40,0,MENUENABLED,"Skok",&StartTekstItem,0,0,0,0};
struct Menu SzukajMenu = {
        &SkokMenu,150,0,56,0,MENUENABLED,"Szukaj",&SzukajItem, 0,0,0,0);
struct Menu EdytorMenu = {
        &SzukajMenu,80,0,56,0,MENUENABLED,"Edycja",&KopiujItem,0,0,0,0);
struct Menu ProjektMenu = {
        &EdytorMenu,0,0,60,0,MENUENABLED,"Projekt",&LadujItem,0,0,0,0);
```

NextSelect: To pole ustawiane jest przez system i informuje, że użytkownik na jednym razem wybrał więcej niż jeden element (więcej na ten temat powiem później). Przy inicjalizacji należy wstawić tam MENUNULL.

A teraz znaczniki pola Flags:

CHECKIT: Każdy element może być używany w dwóch sytuacjach: jako element akcji (action item) lub atrybutu (attribute item). Przykładem elementu akcji jest Brush/Load DeluxePainta. Przykładem elementu atrybutu to choćby Mode/Color tegoż programu. Jeśli dany element ma być elementem atrybutu, należy ustawić ten znacznik.

CHECKED: Element jestelementem atrybutu może znajdować się w dwóch stanach. Wybranie danego elementu powoduje ustawienie znacznika CHECKED. W polu selekcji elementu pojawia się tzw. checkmark - mały znaczek informujący użytkownika o tym, że element jest wybrany. Jeśli dany element jest deselekcjonowany automatycznie (mutual exclude), to zerowany jest również znacznik CHECKED. Jeśli chcemy, aby element był ustawiony od razu przy starcie, możemy ustawić ten znacznik w czasie inicjalizacji.Pamiętać należy, że checkmark ma swoją określoną wielkość. Jeśli nie używamy swojego prywatnego (a można to zrobić wstawiając konkretną strukturę Image w pole CheckMark struktury NewWindow przed otwarciem okna), wielkość znacznika zdefiniowana jest jako CHECKWIDTH lub LOWCHECKWIDTH jeśli pracujemy w niskiej rozdzielczości. Wartość tę lub większą wstawiamy np. do pola LeftEdge struktury IntuiText zawierającej opis naszego elementu.

MENUTOGGLE: działa w połączeniu z CHECKIT. W normalnym trybie wybranie elementu atrybutu powoduje ustawienie wskaźnika CHECKED oraz checkmarku. Skasowanie (deselekcja) możliwa jest tylko z programu - przez programowe wyzerowanie bitu CHECKED lub użycie pola MutualExclude. Jeśli natomiast ustawiony jest bit MENUTOGGLE, dany element działa jako przełącznik. Znaczy to, że każde wybranie elementu powoduje zmianę bitu CHECKIT (i przy okazji MENUTOGGLED) na przeciwny.

MENUTOGGLED: patrz wyżej.

HIGHITEM: element będzie podświetlony i *intuition* ustawi ten znacznik gdy mysz wejdzie w pole selekcji danego elementu.

Następne cztery znaczniki, a właściwie dwubitowe agregaty, określają sposób podświetlenia danego elementu:

HIGHCOMP - wszystkie piksele w polu selekcji zmienią swój kolor na przeciwny;
HIGHBOX - naokoło pola selekcji narysowany będzie prostokąt;
HIGHIMAGE - wyświetlony będzie tekst lub obrazek określony przez pole SelectFill;
HIGHNONE - nic nie będzie podświetlane.

ITEMENABLED: działa tak samo jak MENUENABLED dla menu.

ITEMTEXT: element jest określany przez opis tekstowy, a więc *intuition* będzie spodziewać się, że w polu ItemFill (ewentualnie SelectFill) znajduje się wskaźnik do struktury IntuiText. Jeśli znacznik nie jest ustawiony, zakłada się, że pola te wskazują na strukturę Image.

COMMSEQ: Informuje *intuition*, że element "chodzi" z klawiatury (patrz pole Command). I tu przy deklarowaniu szerokości elementu trzeba uwzględnić szerokość jednego znaku komendy oraz szerokość znaczka "Amiga" (zdeklarowaną jako COMMWIDTH lub LOWCOMMWIDTH).

ISDRAWN: jest ustawiony, gdy wyświetlane jest sub-menu danego elementu. Przykładowa deklaracja listing #2.

Deklarując szerokość menu pamiętać należy o dwóch sprawach.

System może informować program o wybraniu menu na dwa sposoby. Po pierwsze, może nie wnikając w szczegóły wysłać informację o wybraniu konkretnego lub konkretnych elementów po zakończeniu akcji użytkownika, czyli gdy zostawimy mysz w spokoju. Po drugie, może poinformować o próbie uruchomienia menu przez użytkownika. Wówczas decyzję co robić dalej pozostawiamy programowi. Zajmijmy się najpierw tym pierwszym sposobem.

Załóżmy, że okno jest otwarte, menu do okna przydzielone i w polu IDCMPFlags ustawiony jest bit MENUPICK (można to zrobić oczywiście ustawiając go już w strukturze NewWindow lub za pomocą funkcji ModifyIDCMP). Odbierając informację klasy MENUPICK w polu Code dostajemy 16-bitowy numer menu informujący, który element został wybrany. Pięć najmłodszych bitów to numer menu, następne sześć to element i pięć ostatnich to element z submenu. Kombinacja, w której wszystkie bity są ustawione, jest zarezerwowana dla sytuacji, w której użytkownik zrezygnował z wybrania czegokolwiek. Tak więc *intuition* może obsłużyć maksymalnie 31 menu po 63 elementy w każdym oraz po 31 w każdym submenu. Nie wiem jak Wam, ale mnie nie wydaje się to wielkim ograniczeniem.

Trzy makra zdeklarowane w pliku "intuition/intuition.h" służą do obliczania konkretnych interesujących nas wartości. Są to MENUNUM obliczający numer menu, ITEMNUM obliczający numer elementu i SUBNUM obliczający numer elementu w submenu. Odpowiednie stałe NOMENU, NOITEM i NOSUB to kombinacje informujące o zaniechaniu wyboru. Tak więc jeśli np. wybierzemy coś z menu co nam rozwinie submenu, a z tego nie wybierzemy nic, program będzie o tym poinformowany.

LISTING #2

```
Struct MenuItem OpenItem = {
&SaveItem,          /* nastepny na liscie */
0,0,                /* polozenie */
100,10,             /* wymiary */
COMMSEQ             /* dostepny z klawiatury */
ITEMENABLED         /* wlaczony */
ITEMTEXT            /*tekstowy */
HIGHCOMP,           /* sposob podswietlenia */
0,                  /* brak MutualExclude */
&OpenText,          /* wskaznik do opisu */
NULL,               /* wskaznik do podswietlenia - nieuzywany */
'o',                /* klawisz komendy */
NULL,               /* submenu - nieuzywane */
MENUNULL};          /* nastepny wskazany - nieuzywany */
```

Nie zawsze jednak wygodne jest posługiwanie się numerami. Wolelibyśmy znać adres konkretnej struktury MenuItem odnoszącej się do wybranego elementu. Do tego służy funkcja **ItemAddress**. Wywołujemy ją z dwoma parametrami. Pierwszy to adres listy menu, drugi to zwrócony przez **GetMsg** numer, np.:

```
struct MenuItem *MItem;
Item = ItemAddress (&ProjectMenu,Code);
```

Ponieważ w jednej akcji możemy wybrać kilka elementów, *intuition* informuje o tym wstawiając następny numer w pole **NextSelect** odpowiedniej struktury **MenuItem**. Przykładowy sposób obsługi przez program będzie następujący:

```
...
Code = IM->Code;
...
while (Code! = MENUNULL) {
/* obsługa pojedynczego elementu */
Item = ItemAddress (&MyMenu,Code);
Code = Item->NextSelect;
}
```

Jest to sposób zalecany przez wszelkie podręczniki - prosty i jasny.

Jak wspomniałem program może również przejąć kontrolę nad obsługą menu. Co prawda sprowadza się to tylko do zezwolenia na obsługę, ale resztę naprawdę robi za nas *intuition*.

Jeśli ustawiony będzie (wiadomo gdzie) bit MENUVERIFY, system będzie informować nie o zakończeniu akcji, lecz o naciśnięciu prawego klawisza myszy i czekać będzie [illegible] potwierdzenie zanim rozpocznie jakąś akcję. Pozwala to [illegible] np. dokończenie rysowania czegoś na naszym oknie (ponieważ menu nie zostanie wyświetlone zanim prog[illegible] nie wywoła **ReplyMsg**), pozwala również [illegible] zaniechanie obsługi menu. W tym celu w pole **Code**, struktury **IntuiMessage** którą będziemy zwracać systemowi, wstawia się MENUCANCEL - i system już wie, że program nie życzy sobie wyświetlania żadnych menu. W ten sposób działa np. obsługa menu w programie DeluxePaint:

```
if (IM->Class= =MENUVERIFY) {
    if (IM->MouseY> Window->BorderTop)
    IM->Code= MENUCANCEL;
    ReplyMsg(IM);
    continue;
    }
```

Tym razem przykładowy program będzie tylko jeden, al[illegible] [illegible] to duży. Mam nadzieję, że po przeanalizowaniu [illegible] każdy będzie w stanie napisać własne programy obsługi menu.

Wyjaśnienia wymaga kilka spraw:

Sprawdzenie przez MENUVERIFY jest bardzo wygodne, lecz [illegible] jedną wadę. Jeśli robimy to bezpośrednio, możemy spotkać się z przykrą niespodzianką: otóż nie działa również prawidłowo wybieranie menu z klawiatury. Dlatego dodatkowo sprawdzam pole **Qualifier** po to by sprawdzić, czy menu było uruchamia[illegible] klawiszem myszy (ustawiony bit IEQUALIFIER_RBUTTON). Jeśli nie - to nie trzeba nic kasować, menu powinno być obsłużone.

Bardzo wygodne byłoby zdeklarowanie funkcji, która zwraca programowi informację o pojedynczej akcji użytkownika. W językach programowania dużo wyższego poziomu (jak Icon) moglibyśmy użyć do tego instrukcji **suspend** działającej podobnie jak **return**, lecz przy następnym wywołaniu funkcji jej działanie nie jest rozpoczynane od początku, lecz wznawiane od miejsca za instrukcją **suspend**. Ponieważ C takiej instrukcji nie posiada, pokażę, jak można w prosty sposób zasymulować jej działanie.

Suspend w funkcji użyty jest dwukrotnie: dla akcji jak wyżej oraz po to, by uchronić się przed nieprzyjemną sytuacją przy rysowaniu kół i elips, gdy szybko przelecimy myszą w inne miejsce, a Amiga tymczasem rysuje kolejne elipsy we wszystkich miejscach, gdzie po drodze przerwanie zastała mysz. Dlatego po odebraniu wiadomości klasy MOUSEMOVE program sprawdza, czy następną wiadomością jest również MOUSEMOVE. Jeśli tak - uwzględnia tylko następną wiadomość.

Oczywiście program nie jest żadnym programem użytkowym! Proponuję jednak wpisanie go na swój dysk, gdyż w następnym odcinku programem demonstracyjnym będzie jego modyfikacja.

Życzę udanych eksperymentów.

Bohdan R. Rau

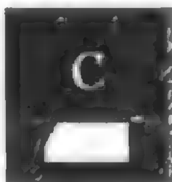
PS. To już naprawdę przedostatni odcinek z serii "czytanie myszy". Do obróbki pozostały [illegible] (przynajmniej [illegible] tym etapie) tylko gadgety, ale o tym [illegible] miesiąc. Przygotujcie się [illegible] naprawdę dużą porcję informacji!

```
#include <intuition/intuition.h>
#include <exec/memory.h>
#include <functions.h>

struct IntuitionBase *IntuitionBase;
struct GfxBase *GfxBase;
struct Window *Window;
struct Screen *Screen;
struct TextAttr TAttr={(STRPTR)"topaz.font",8,0,FPF_ROMFONT};
struct TextAttr
    BTAttr={(STRPTR)"topaz.font",8,FSF_BOLD,FPF_ROMFONT};
struct TextAttr
    ITAttr={(STRPTR)"topaz.font",8,FSF_ITALIC,FPF_ROMFONT};
struct TextAttr
  UTAttr={(STRPTR)"topaz.font",8,FSF_UNDERLINED,FPF_ROMFONT};

UWORD Mask0[]={
    0xffff,0xfffc,0xffff,0xfffc,0xc000,0x000c,0xc000,0x000c,
    0xc000,0x000c,0xc000,0x000c,0xc000,0x000c,0xc000,0x000c,
    0xc000,0x000c,0xc000,0x000c,0xc000,0x000c,0xc000,0x000c,
    0xc000,0x000c,0xc000,0x000c,0xffff,0xfffc,0xffff,0xfffc,
    0xffff,0xfffc,0xffff,0xfffc,0xc000,0x000c,0xc000,0x000c,
    0xc000,0x000c,0xc000,0x000c,0xc000,0x000c,0xc000,0x000c,
    0xc000,0x000c,0xc000,0x000c,0xc000,0x000c,0xc000,0x000c,
    [illegible],0x000c,0xc000,0x000c,0xffff,0xfffc,0xffff,0xfffc};
UWORD Mask1[]={
    0xffff,0xfffc,0xffff,0xfffc,0xffff,0xfffc,0xffff,0xfffc,
    0xffff,0xfffc,0xffff,0xfffc,0xffff,0xfffc,0xffff,0xfffc,
    0xffff,0xfffc,0xffff,0xfffc,0xffff,0xfffc,0xffff,0xfffc,
    0xffff,0xfffc,0xffff,0xfffc,0xffff,0xfffc,0xffff,0xfffc,
    0xffff,0xfffc,0xffff,0xfffc,0xc000,0x000c,0xc000,0x000c,
    0xc000,0x000c,0xc000,0x000c,0xc000,0x000c,0xc000,0x000c,
    0xc000,0x000c,0xc000,0x000c,0xc000,0x000c,0xc000,0x000c,
    0xc000,0x000c,0xc000,0x000c,0xffff,0xfffc,0xffff,0xfffc};
UWORD Mask2[]={
    0xffff,0xfffc,0xffff,0xfffc,0xc000,0x000c,0xc000,0x000c,
    0xc000,0x000c,0xc000,0x000c,0xc000,0x000c,0xc000,0x000c,
    0xc000,0x000c,0xc000,0x000c,0xc000,0x000c,0xc000,0x000c,
    0xc000,0x000c,0xc000,0x000c,0xffff,0xfffc,0xffff,0xfffc,
    0xffff,0xfffc,0xffff,0xfffc,0xffff,0xfffc,0xffff,0xfffc,
    0xffff,0xfffc,0xffff,0xfffc,0xffff,0xfffc,0xffff,0xfffc,
    0xffff,0xfffc,0xffff,0xfffc,0xffff,0xfffc,0xffff,0xfffc,
    0xffff,0xfffc,0xffff,0xfffc,0xffff,0xfffc,0xffff,0xfffc};
UWORD Mask3[]={
    0x0000,0x0000,0x3fff,0xfff0,0x3fff,0xfff0,0x3fff,0xfff0,
    0x3fff,0xfff0,0x3fff,0xfff0,0x3fff,0xfff0,0x3fff,0xfff0,
    0x3fff,0xfff0,0x3fff,0xfff0,0x3fff,0xfff0,0x3fff,0xfff0,
    0x3fff,0xfff0,0x3fff,0xfff0,0x3fff,0xfff0,0x0000,0x0000,
    0xffff,0xfffc,0xffff,0xfffc,0xffff,0xfffc,0xffff,0xfffc,
    0xffff,0xfffc,0xffff,0xfffc,0xffff,0xfffc,0xffff,0xfffc,
    0xffff,0xfffc,0xffff,0xfffc,0xffff,0xfffc,0xffff,0xfffc,
    0xffff,0xfffc,0xffff,0xfffc,0xffff,0xfffc,0xffff,0xfffc};

/*   wybor koloru     */
```



```
struct Image Kol0Pic={
    LOWCHECKWIDTH,2,30,16,2,0,3,0,0};
struct Image Kol1Pic={
    LOWCHECKWIDTH,2,30,16,2,0,3,0,0};
struct Image Kol2Pic={
    LOWCHECKWIDTH,2,30,16,2,0,3,0,0};
struct Image Kol3Pic={
    LOWCHECKWIDTH,2,30,16,2,0,3,0,0};

struct MenuItem Kol3Item={
    0,LOWCHECKWIDTH+30,20,LOWCHECKWIDTH+30,20,
    ITEMENABLED|HIGHCOMP|CHECKIT,
    1|2|4,(APTR)&Kol3Pic,0,0,0,MENUNULL};
[redacted] MenuItem [redacted]={
    &Kol3Item,0,20,LOWCHECKWIDTH+30,20,
    ITEMENABLED|HIGHCOMP|CHECKIT,
    1|2|8,(APTR)&Kol2Pic,0,0,0,MENUNULL};
struct MenuItem Kol1Item={
    &Kol2Item,LOWCHECKWIDTH+30,0,LOWCHECKWIDTH+30,20,
    ITEMENABLED|HIGHCOMP|CHECKIT|CHECKED,
    1|4|8,(APTR)&Kol1Pic,0,0,0,MENUNULL};
struct MenuItem Kol0Item={
    &Kol1Item,0,0,LOWCHECKWIDTH+30,20,
    ITEMENABLED|HIGHCOMP|CHECKIT,
    2|4|8,(APTR)&Kol0Pic,0,0,0,MENUNULL};

/*    wybor stylu:          */

struct IntuiText UnderText={
    0,1,JAM2,LOWCHECKWIDTH,1,&UTAttr,(STRPTR)"Podkreslony",0};
struct IntuiText BoldText={
    0,1,JAM2,LOWCHECKWIDTH,1,&BTAttr,(STRPTR)"Tlusty",0};
struct IntuiText ItalText={
    0,1,JAM2,LOWCHECKWIDTH,1,&ITAttr,(STRPTR)"Kursywa",0};
struct IntuiText NormalText={
    0,1,JAM2,LOWCHECKWIDTH,1,&TAttr,(STRPTR)"Normalny",0};

struct MenuItem UnderItem={
    0,60,30,102,10,
    ITEMTEXT|ITEMENABLED|HIGHCOMP|CHECKIT|MENUTOGGLE,
    1,(APTR)&UnderText,0,0,0,MENUNULL};
struct MenuItem BoldItem={
    &UnderItem,60,20,102,10,
    ITEMTEXT|ITEMENABLED|HIGHCOMP|CHECKIT|MENUTOGGLE,
    1,(APTR)&BoldText,0,0,0,MENUNULL};
struct MenuItem ItalItem={
    &BoldItem,60,10,102,10,
    ITEMTEXT|ITEMENABLED|HIGHCOMP|CHECKIT|MENUTOGGLE,
    1,(APTR)&ItalText,0,0,0,MENUNULL};
struct MenuItem NormalItem={
    &ItalItem,60,0,102,10,
    ITEMTEXT|ITEMENABLED|HIGHCOMP|CHECKIT|CHECKED,
    2|4|8,(APTR)&NormalText,0,0,0,MENUNULL};

/* sposob rysowania */

struct IntuiText StylText={
    0,1,JAM2,0,1,&TAttr,(STRPTR)"Styl pisania",0};
struct IntuiText PelnyText={
    0,1,JAM2,LOWCHECKWIDTH,1,&TAttr,(STRPTR)"Zamaluj",0};
struct IntuiText MalujText={
    0,1,JAM2,LOWCHECKWIDTH,1,&TAttr,(STRPTR)"Maluj",0};
struct IntuiText TekstText={
    0,1,JAM2,LOWCHECKWIDTH,1,&TAttr,(STRPTR)"Tekst",0};
struct IntuiText PunktText={
    0,1,JAM2,LOWCHECKWIDTH,1,&TAttr,(STRPTR)"Punkt",0};
struct IntuiText LiniaText={
    0,1,JAM2,LOWCHECKWIDTH,1,&TAttr,(STRPTR)"Linia",0};
struct IntuiText KoloText={
    0,1,JAM2,LOWCHECKWIDTH,1,&TAttr,(STRPTR)"Kolo",0};
struct IntuiText ElipText={
    0,1,JAM2,LOWCHECKWIDTH,1,&TAttr,(STRPTR)"Elipsa",0};
struct IntuiText KwadrText={
    0,1,JAM2,LOWCHECKWIDTH,1,&TAttr,(STRPTR)"Prostokat",0};

struct MenuItem StylItem={
    0,0,80,100,10,
    ITEMTEXT|ITEMENABLED|HIGHCOMP,
    0,(APTR)&StylText,0,0,&NormalItem,MENUNULL};
struct MenuItem PelnyItem={
    &StylItem,0,70,100,10,
    ITEMTEXT|ITEMENABLED|HIGHCOMP|CHECKIT|MENUTOGGLE,
    0,(APTR)&PelnyText,0,0,0,MENUNULL};
struct MenuItem KwadrItem={
    &PelnyItem,0,60,100,10,
    ITEMTEXT|ITEMENABLED|HIGHCOMP|CHECKIT,
    1|2|4|8|16|32,(APTR)&KwadrText,0,0,0,MENUNULL};
struct MenuItem KoloItem={
    &KwadrItem,0,50,100,10,
    ITEMTEXT|ITEMENABLED|HIGHCOMP|CHECKIT,
    1|2|4|8|16|64,(APTR)&KoloText,0,0,0,MENUNULL};
struct MenuItem ElipItem={
    &KoloItem,0,40,100,10,
    ITEMTEXT|ITEMENABLED|HIGHCOMP|CHECKIT,
    1|2|4|8|32|64,(APTR)&ElipText,0,0,0,MENUNULL};
struct MenuItem MalujItem={
    &ElipItem,0,30,100,10,
    ITEMTEXT|ITEMENABLED|HIGHCOMP|CHECKIT,
    1|2|4|16|32|64,(APTR)&MalujText,0,0,0,MENUNULL};
struct MenuItem TekstItem={
    &MalujItem,0,20,100,10,
    ITEMTEXT|ITEMENABLED|HIGHCOMP|CHECKIT,
    1|2|8|16|32|64,(APTR)&TekstText,0,0,0,MENUNULL};
struct MenuItem PunktItem={
    &TekstItem,0,10,100,10,
    ITEMTEXT|ITEMENABLED|HIGHCOMP|CHECKIT|CHECKED,
    1|4|8|16|32|64,(APTR)&PunktText,0,0,0,MENUNULL};
struct MenuItem LiniaItem={
    &PunktItem,0,0,100,10,
    ITEMTEXT|ITEMENABLED|HIGHCOMP|CHECKIT,
    2|4|8|16|32|64,(APTR)&LiniaText,0,0,0,MENUNULL};

/* Projekt */

struct IntuiText KonczText={
    0,1,JAM2,0,1,&TAttr,(STRPTR)"Koniec",0};
struct IntuiText CofnijText={
    0,1,JAM2,0,1,&TAttr,(STRPTR)"Cofnij",0};
struct IntuiText NowyText={
    0,1,JAM2,0,1,&TAttr,(STRPTR)"Nowy rys.",0};

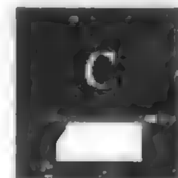
struct MenuItem KonczItem={
    0,0,20,100,10,
    ITEMTEXT|ITEMENABLED|HIGHCOMP|COMMSEQ,
    0,(APTR)&KonczText,0,'q',0,MENUNULL};
struct MenuItem CofnijItem={
    &KonczItem,0,10,100,10,
    ITEMTEXT|ITEMENABLED|HIGHCOMP|COMMSEQ,
    0,(APTR)&CofnijText,0,'c',0,MENUNULL};
struct MenuItem NowyItem={
    &CofnijItem,0,0,100,10,
    ITEMTEXT|ITEMENABLED|HIGHCOMP,
    0,(APTR)&NowyText,0,0,0,MENUNULL};

/* Deklaracja menu */

struct Menu KolorMenu={
    0,160,0,65,0,MENUENABLED,"Kolor",&Kol0Item};
struct Menu RysujMenu={
    &KolorMenu,80,0,65,0,MENUENABLED,"Rysuj",&LiniaItem};
struct Menu ProjektMenu={
    &RysujMenu,0,0,65,0,MENUENABLED,"Projekt",&NowyItem};

struct NewWindow nwindow={
    0,0,320,256,2,1,
    MOUSEBUTTONS|MOUSEMOVE|MENUVERIFY|MENUPICK
                                          |VANILLAKEY,
    ACTIVATE|GIMMEZEROZERO|REPORTMOUSE,0,0,
    (STRPTR)"Graphic-Haft",
    0,0,0,0,0,0,CUSTOMSCREEN};
struct NewScreen [redacted]een={
    0,0,320,256,2,2,1,0,WBENCHSCREEN,&TAttr,0,0,0};
struct RastPort XPort;
struct BitMap XMap;
struct AreaInfo AI;
struct TmpRas TR;
char *TB;
SHORT AMem[160];

OpenAll()
{
        if (!(IntuitionBase=(struct IntuitionBase *)
          OldOpenLibrary("intuition.library"))) clean(100);
        if (!(GfxBase=(struct GfxBase *)
     OldOpenLibrary("graphics.library"))) clean(100);
        if (!(Screen=OpenScreen(&nscreen))) clean(100);
        nwindow.Screen=Screen;
        if (!(Window=OpenWindow(&nwindow))) clean(100);
```



```
        if (!(Kol0Pic.ImageData=(USHORT *)
    AllocMem(128,MEMF_CHIP|MEMF_CLEAR))) clean(100);
        if (!(Kol1Pic.ImageData=(USHORT *)
    AllocMem(128,MEMF_CHIP|MEMF_CLEAR))) clean(100);
        if (!(Kol2Pic.ImageData=(USHORT *)
    AllocMem(128,MEMF_CHIP|MEMF_CLEAR))) clean(100);
        if (!(Kol3Pic.ImageData=(USHORT *)
    AllocMem(128,MEMF_CHIP|MEMF_CLEAR))) clean(100);
        InitRastPort(&XPort);
        InitBitMap(&XMap,2,320,256);
        XPort.BitMap=&XMap;
        if (!(XMap.Planes[0]=AllocRaster(320,256))) clean(100);
        if (!(XMap.Planes[1]=AllocRaster(320,256))) clean(100);
        if (!(TB=(char *)AllocRaster(320,256))) clean(100);
        InitTmpRas(&TR,TB,RASSIZE(320,200));
        InitArea(&AI,AMem,64);
        Window->RPort->TmpRas=&TR;
        Window->RPort->AreaInfo=&AI;
        CopyMem((char *)Mask0,(char *)Kol0Pic.ImageData,128);
        CopyMem((char *)Mask1,(char *)Kol1Pic.ImageData,128);
        CopyMem((char *)Mask2,(char *)Kol2Pic.ImageData,128);
        CopyMem((char *)Mask3,(char *)Kol3Pic.ImageData,128);
        SetMenuStrip(Window,&ProjektMenu);
        Zapamietaj();
        SetAPen(Window->RPort,1);
        SetDrMd(Window->RPort,JAM1);
}

clean(n) int n;
{
        if (Kol3Pic.ImageData) FreeMem(Kol3Pic.ImageData,128);
        if (Kol2Pic.ImageData) FreeMem(Kol2Pic.ImageData,128);
        if (Kol1Pic.ImageData) FreeMem(Kol1Pic.ImageData,128);
        if (Kol0Pic.ImageData) FreeMem(Kol0Pic.ImageData,128);
        if (XMap.Planes[0]) FreeRaster(XMap.Planes[0],320,256);
        if (XMap.Planes[1]) FreeRaster(XMap.Planes[1],320,256);
        if (TB) FreeRaster(TB,320,256);
        if (Window) {
    Window->RPort->TmpRas=0;
    Window->RPort->AreaInfo=0;
    ClearMenuStrip(Window);
    CloseWindow(Window);
    }
        if (Screen) CloseScreen(Screen);
        if (GfxBase) CloseLibrary(GfxBase);
        if (IntuitionBase) CloseLibrary(IntuitionBase);
        exit(n);
}

#define R_PUNKTY 0
#define R_LINIE 1
#define R_TEKST 3
#define R_MALUJ 4
#define R_KWADR 5
#define R_KOLA 6
#define R_ELIPSY 7

ULONG Class;USHORT Code;
int ButtonPos,Pozycja,WMouseX,WMouseY;
int Rysowanie=R_PUNKTY;
int DrawX,DrawY,RadX,RadY;
#define zamalowany (PelnyItem.Flags & CHECKED)
#define wybrany(x) (x.Flags & CHECKED)

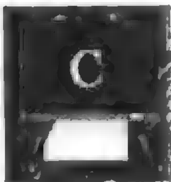
main()
{
        char tc;
        OpenAll();
        for (;;) {
    GetMessage();
    if (Class==VANILLAKEY && Rysowanie==R_TEKST && Pozycja) {
        if (Code==13 /* ENTER */) goto nowa_linia;
        Move(Window->RPort,DrawX,DrawY+6);
        tc=Code;
        Text(Window->RPort,&tc,1);
        DrawX+=8;
        if (DrawX+8>Window->GZZWidth) {
nowa_linia: DrawX=0;
        DrawY+=8;
        if (DrawY+8>=Window->GZZHeight) DrawY=0;
        }
        continue;
        }
    if (Class==MOUSEMOVE) {
```

```
        if(ButtonPos)DoDraw();
        continue;
        }
    if (Class==MOUSEBUTTONS) {
     if (Code==SELECTUP) {
     ButtonPos=0;
     StopDraw();
     continue;
     }
     if (Code==SELECTDOWN) {
     ButtonPos=1;
     if (WMouseX>=0 && WMouseX<Window->GZZWidth &&
        WMouseY>=0 && WMouseY<Window->GZZHeight) StartDraw();
     continue;
     }
     continue;
     }
    if (Class==MENUPICK) {
     ButtonPos=0;
     switch(MENUNUM(Code)) {
     case 0: Projekt();break;
     case 1: Rysuj();break;
     case 2: Kolor();break;
     }
     continue;
     }
    }
}

StartDraw()
{
      DrawX=WMouseX;
      DrawY=WMouseY;
      Zapamietaj();
      switch(Rysowanie) {
   case R_KOLA:
   case R_ELIPSY:
   case R_PUNKTY:
   case R_LINIE:
   case R_KWADR:
   WritePixel(Window->RPort,WMouseX,WMouseY);
   Move(Window->RPort,WMouseX,WMouseY);
   return;
   case R_MALUJ:
   Flood(Window->RPort,1,WMouseX,WMouseY);
   return;
   case R_TEKST:
   DrawX=WMouseX;DrawY=WMouseY;Pozycja=1;
   if (DrawX+8>Window->GZZWidth) DrawX=Window->GZZWidth-8;
   if (DrawY+8>Window->GZZHeight) DrawY=Window->GZZHeight-8;
   return;
   }
}

DoDraw()
{
     switch(Rysowanie) {
   case R_PUNKTY:
   if (zamalowany) Draw(Window->RPort,WMouseX,WMouseY);
   else WritePixel(Window->RPort,WMouseX,WMouseY);
   return;
   case R_LINIE:
   Odtworz();
   Move(Window->RPort,DrawX,DrawY);
   Draw(Window->RPort,WMouseX,WMouseY);
   return;
   case R_KWADR:
   Odtworz();Kwadrat();return;
   case R_ELIPSY:
   case R_KOLA:
   Odtworz();Kolo();return;
   }
}

StopDraw()
{
    switch(Rysowanie) {
   case R_LINIE:
   case R_KWADR:
   case R_ELIPSY:
   case R_KOLA:
   DoDraw();
   }
}
```



```
Kolo()
{
      RadX=abs(WMouseX-DrawX);
      if (DrawX-RadX<0) RadX=DrawX;
      if (DrawX+RadX>=Window->GZZWidth)
      RadX=Window->GZZWidth-DrawX-1;
      if (Rysowanie==R_KOLA) RadY=RadX;else
      RadY=abs(WMouseY-DrawY);
      if (DrawY-RadY<0) RadY=DrawY;
      if (DrawY+RadY>=Window->GZZHeight)
      RadY=Window->GZZHeight-DrawY-1;
      if (Rysowanie==R_KOLA && RadY<RadX) RadX=RadY;
      if (zamalowany) {
    AreaEllipse(Window->RPort,DrawX,DrawY,RadX,RadY);
    AreaEnd(Window->RPort);
    }
      [redacted] DrawEllipse(Window->RPort,DrawX,DrawY,RadX,RadY);
}

Kwadrat()
{
      int x1,x2,y1,y2;
      if (zamalowany) {
    if (DrawX<WMouseX) {x1=DrawX;x2=WMouseX;}
    [redacted] {x2=DrawX;x1=WMouseX;}
    if (DrawY<WMouseY) {y1=DrawY;y2=WMouseY;}
    else {y2=DrawY;y1=WMouseY;}
    RectFill(Window->RPort,x1,y1,x2,y2);
    }
      else {
    Move(Window->RPort,DrawX,DrawY);
    Draw(Window->RPort,DrawX,WMouseY);
    Draw(Window->RPort,WMouseX,WMouseY);
    Draw(Window->RPort,WMouseX,DrawY);
    Draw(Window->RPort,DrawX,DrawY);
    }
}

Zapamietaj()
{
      ClipBlit(
        Window->RPort,0,0,&XPort,0,0,
    Window->GZZWidth,Window->GZZHeight,0xc0);
}

Odtworz()
{
    ClipBlit(
    &XPort,0,0,Window->RPort,0,0,
    Window->GZZWidth,Window->GZZHeight,0xc0);
}

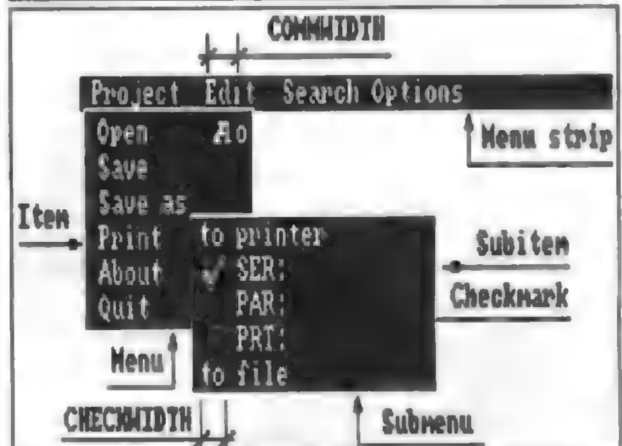
Projekt()
{
      switch(ITEMNUM(Code)) {
    case 0:Zapamietaj();SetRast(Window->RPort,0);return;
    [redacted] 1:Odtworz();return;
    case 2:clear(0);
    }
}

Rysuj()
{
    if (ITEMNUM(Code)==6) return Styl();
    if (wybrany(LiniaItem)) Rysowanie=R_LINIE;else
    if (wybrany(PunktItem)) Rysowanie=R_PUNKTY;else
    if (wybrany(KwadrItem)) Rysowanie=R_KWADR;else
    if (wybrany(MalujItem)) Rysowanie=R_MALUJ;else
    if (wybrany(KoloItem))  Rysowanie=R_KOLA;else
    if (wybrany(ElipItem))  Rysowanie=R_ELIPSY;else
    if (wybrany(TekstItem)) {Rysowanie=R_TEKST;Pozycja=0;}
}

Styl()
{
    UBYTE SM;
    SM=0;
    if (wybrany(BoldItem))SM|=FSF_BOLD;
    if (wybrany(ItalItem))SM|=FSF_ITALIC;
    if (wybrany(UnderItem))SM|=FSF_UNDERLINED;
    SetSoftStyle(Window->RPort,SM,255);
}
```

```
Kolor()
{
    if (wybrany(Kol0Item)) SetAPen(Window->RPort,0);else
    if (wybrany(Kol1Item)) SetAPen(Window->RPort,1);else
    if (wybrany(Kol2Item)) SetAPen(Window->RPort,2);else
    if (wybrany(Kol3Item)) SetAPen(Window->RPort,3);
}

#define suspend(n,l) {susp=n;return;l:susp=0;}
GetMessage()
{
        [redacted] struct IntuiMessage *IM;
        [redacted] MenuItem *MI;
        static int susp=0; /* emulator suspend - zmienna kontrolna */
        static USHORT _Code;
        if (susp==1) goto z1;
        if (susp==2) goto z2;
        for (;;) {
      while (!(IM=(struct IntuiMessage *)
      GetMsg(Window->UserPort)))
      Wait(1<<Window->UserPort->mp_SigBit);
obrobka:
      Class=IM->Class;
      Code=IM->Code;
      WMouseX=IM->MouseX-Window->BorderLeft;
      WMouseY=IM->MouseY-Window->BorderTop;
      if (Class==MENUVERIFY) {
        if (WMouseY>0 && (IM->Qualifier & IEQUALIFIER_RBUTTON))
        IM->Code=MENUCANCEL;
        ReplyMsg((struct Message *)IM);
        continue;
        }
      ReplyMsg((struct Message *)IM);
      if (Class==MOUSEMOVE) {
        if (WMouseX<0 || WMouseX>=Window->GZZWidth) continue;
        if (WMouseY<0 || WMouseY>=Window->GZZHeight) continue;
        IM=(struct IntuiMessa[redacted] *)GetMsg(Window->UserPort);
        if (!IM) return;
        if (IM->Class!=MOUSEMOVE) suspend(1,z1);
        goto obrobka;
        }
      if (Class==MENUPICK) {
        for (;;) {
        if (Code==MENUNULL) break;
        _Code=Code;
        suspend(2,z2);
        MI=ItemAddress(&ProjektMenu,_Code);
        if (!MI) break;
        Code=MI->NextSelect;
        }
        continue;
        }
      if (Class==MOUSEBUTTONS) {
        if (Code!=SELECTDOWN [redacted] Code!=SELECTUP) continue;
        return;
        }
```



rys. 1: Wygląd rozwiniętego menu





PREHISTORIC TALE

THALION

Erupcje wulkanów i trzęsienia ziemi spowodują niedługo całkowitą śmierć dinozaurów. Zupełna zagłada tych gadów stanie się ogromną stratą dla ludzkości, która nie będzie mogła poznać zwyczajów oraz budowy tych stworzeń. Dinozaury tak ciekawiły naukowców, że starali się oni zdobyć jak najwięcej informacji na temat tych "tajemniczych" istot. Gwałtowny rozwój techniki umożliwił zbudowanie maszyny czasu. Natychmiast zrodził się pomysł wysłania w czasy prehistoryczne ekspedycji, której celem byłoby ocalenie dinozaurów przed zagładą. Specjalna grupa ludzi, na której czele stanął Cronos Warchild, otrzymała zadanie przeniesienia jaj dinozaurów z obszarów zagrożonych kataklizmami do bardziej spokojnych rejonów ówczesnego świata. W postać Cronosa wcielasz się Ty, drogi graczu. Zostajesz teleportowany i pojawiasz się na pierwszej planszy, gdzie pośród kamieni, różnych drabin i półek, znajdują się jaja dinozaurów. Możesz poruszać się, skacząc z półki na półkę lub wspinając się po drabinkach. Musisz zebrać tyle jaj, ile tylko zdołasz i przenieść je za pomocą teleportu, zanim zdołają się z nich wykluć malutkie dinozaury. Możesz nieść jednocześnie tylko trzy jaja, chyba, że znalazłeś baterię, która dodaje Ci siły. Czasami na Twojej drodze staną rozwścieczeni rodzice małego dinozaura, którym nie można wytłumaczyć, że staramy się ocalić ich przyszłą pociechę. Wówczas należy, bądź wypuścić jedno jajko, bądź spróbować zabić przeciwnika celnie zrzuconym kamieniem. Po zebraniu wszystkich jajek na danym poziomie przenosisz się na kolejną planszę, gdzie ponawiasz swoje wysiłki. Gra zarówno od strony pomysłu, jak i jego realizacji, jest słaba. Przyzwyczailiśmy się do wspaniałej grafiki i ciekawych efektów dźwiękowych przy grach platformowych, w których trudno o jakiś zaskakujący scenariusz. A PREHISTORIC TALE jest krokiem w tył w świecie platformówek.

Ocena	*niedostateczna plus*					
Grafika	*					
Dźwięk	*					
Pomysł	*					
Motywacja	*					
OGÓLNIE	*					



WINGS OF FURY

Walki toczone podczas Drugiej Wojny Światowej przez żołnierzy amerykańskich i japońskich na wyspach Pacyfiku były jej najkrwawszymi epizodami. Po załamaniu się ofensywy japońskiej, Amerykanie powoli odzyskiwali utracone wyspy i archipelagi. Niezwykła specyfika działań w tym rejonie przyczyniła się do wytworzenia niespotykanej dotąd taktyki walki, znanej dziś pod nazwą "żabich skoków". Polegała ona na tym, że proces zdobywania kolejnych wysp składał się z trzech etapów: morderczych nalotów, desantu oraz umacniania zdobytej pozycji. W ten sposób siły amerykańskie krok po kroku odzyskiwały utracone na rzecz Japończyków rejony Pacyfiku. Scenerię tę miała zobrazować nam gra pt. WINGS OF FURY. Niestety, jest ona tak słaba, że mogłaby stanowić przykład, jak nie powinna wyglądać gra komputerowa; zwłaszcza, na tak wspaniały komputer, jakim niewątpliwie jast nasza AMIGA. Do wyboru mamy siedem misji różniących się tylko ilością wysp, które mamy zbombardować i nagromadzeniem okrętów do storpedowania (to jest naszym głównym zadaniem). Lecimy samolotem uzbrojonym w 30 bomb, 15 rakiet, jedną torpedę, strzelając do stanowisk artylerii, piechoty japońskiej oraz okrętów i samolotów wroga. Po wyczerpaniu amunicji lub też w celu usunięcia uszkodzeń wracamy na nasz lotniskowiec, gdzie ponownie jesteśmy uzbrajani i po krótkim odpoczynku wracamy do akcji. Grafika jest wprost beznadziejna. Wolno przesuwający się, monotonny krajobraz jest w stanie uśpić każdego, nawet najmniej wybrednego gracza. Dźwięk, to tylko świst spadających bomb i szum karabinu maszynowego. Uważam, że jeżeli ktoś chce się zniechęcić do gier komputerowych lub zerwać z tym nałogiem, powinien koniecznie zagrać w WINGS OF FURY.

Ocena	*niedostateczna*					
Grafika						
Dźwięk						
Pomysł	*					
Motywacja	*					
OGÓLNIE	*					



DAS BOOT

Praktycznie od początków swojego istnienia Wielka Brytania przykładała wielką wagę do panowania na morzu. Od stuleci posiadała najpotężniejszą flotę wojenną na Świecie. Wyspiarskie położenie Anglii wymuszało konieczność zabezpieczenia szlaków morskich, co również przyczyniało się do wzrostu zainteresowania flotą wojenną. Podczas Drugiej Wojny Światowej obsesją wodza Trzeciej Rzeszy, Adolfa Hitlera, było zawładnięcie całym Światem, a co za tym idzie i "morzem". Niemcy wiedząc, że długo nie zdołają zniwelować przewagi państw sprzymierzonych w siłach nawodnych, położyli główny nacisk na rozwój swoich sił podwodnych. Wilcze stada stały się prawdziwym postrachem morskich szlaków. Dzięki lepszym okrętom podwodnym, Niemcy zaczęli poważnie zagrażać sprzymierzonym. Służba na U-BOOTACH była uważana za najwyższy zaszczyt. I właśnie program pt. DAS BOOT przenosi nas w świat walki o panowanie na morzach i oceanach. Jest to bardzo dobra symulacja niemieckiej łodzi podwodnej. Do wyboru otrzymujemy trzy najczęściej spotykane w głębinach oceanów okręty niemieckie: Type VII, Type VII-B oraz NEW VII-C. Możemy uczestniczyć w kilku różnych misjach. Aby zobaczyć, co naprawdę czeka nas podczas bojowych patroli proponuję zaliczyć różnorodne ćwiczenia, które dadzą nam obraz, czego mamy się spodziewać podczas wypełniania misji. Ćwiczenia dzielą się na dwa etapy: nawodne i podwodne. Na wiele różnych sposobów możemy ustalać warunki walki (poziom gry, niezawodność torped, szybkość usuwania uszkodzeń, wyposażenie u-boota). Gra wspaniale prezentuje się od strony graficznej. Sylwetki wrogich statków i okrętów są bardzo wyraźne. Niezłe są efekty dźwiękowe. Jednak największym atutem tej gry jest bardzo realistyczne przedstawienie warunków podmorskiej rywalizacji.

Ocena	*super!*					
Grafika	*	*	*	*	*	
Dźwięk	*	*	*	*	*	
Pomysł	*	*	*	*	*	
Motywacja	*	*	*	*	*	*
OGÓLNIE	*	*	*	*	*	*



REALMS

Ktoś ma małe dwupokojowe mieszkanie, inny całe królestwo. Jednak zwykle przychodzi moment, że obaj stoją przed tym samym problemem: z tymi okropnymi sąsiadami nie można dojść do ładu. Jak można się domyślać, w grze Realms stworzonej przez programistów firmy Virgin Games nie chodzi o mieszkanie, lecz o królestwo. Producenci gry dali duży wybór krain. Możemy wybrać kraj krasnali, amazonek, wikingów, orków, barbarzyńców, czy nawet elfów. Gdy już podejmiemy decyzję, jaką rasą chcemy władać, komputer przejmuje sterowanie wszystkimi innymi, którzy stają się sąsiadami i konkurentami.

Wszyscy dążą do tego samego: opanować terytoria sąsiadów. Zasady gry przypominają bardzo "Powermongera". Podobnie jak w tamtej grze, Realms oferuje małą mapę ukazującą teren o który toczy się walka oraz powiększony trójwymiarowy wycinek obejmujący wybrany fragment pola gry. W tym miejscu przedstawiana jest większość akcji. Jedynie w bitwach, w których uczestniczy większa ilość jednostek, pojawia się specjalny ekran walki. Gra oferuje do wyboru siedem scenariuszy, czyli więcej niż "Powermonger".

Co należy do naszych obowiązków jako króla? Otóż wszystko. Należy zbierać podatki od ludności, w zamian ponosząc niezbędne wydatki na zapewnienie jej dostatecznej ilości pożywienia i zabezpieczenia opieki zdrowotnej. Od przygotowywania terenów pod zabudowę i stawiania murów obronnych także nie można się wymigać. Ponieważ trudno jest prowadzić ekspansję terytorialną nie używając sił zbrojnych, musimy formować i opłacać coraz liczniejsze armie niezbędne do strzeżenia królestwa i poszerzania jego granic.

W "Realms" programiści położyli większy nacisk na elementy strategii niż w "Powermonger". Można tworzyć różne formacje wojskowe i uzbrajać je na kilka sposobów. Sądzę, że w celu przybliżenia gry naszym Czytelnikom, dobrze jest podać parę szczegółów dotyczących jakości gry od strony technicznej. Sterowanie odbywa się wyłącznie za pomocą myszy. Jest ono zorganizowane bardzo przejrzyście i nietrudno się zorientować w czym rzecz już po krótkim obcowaniu z grą. Programiści włożyli wiele trudu w utrakcyjniające grę sekwencje wstępne i końcowe oraz w plansze uzbrajania armii itp. Grafika planszy głównej, na której dokonujemy większości operacji jest mniej szczegółowa niż w "Powermonger". Grze towarzyszy niezła muzyka i efekty. Daje się zauważyć parę drobnych niedociągnięć, jak np. wpływające w głąb lądu okręty, itd. W celu ułatwienia życia potencjalnym władcom, zamieszczamy poniżej opis funkcji poszczególnych ikon. Niestety, nie będzie on ilustrowany obrazkami z gry, gdyż programiści chyba przewidzieli, że będziemy chcieli opisać ich dzieło i zabezpieczyli swój produkt przed operacjami Action Replay`em. Po uruchomieniu freezera i wpisaniu rozkazu poszukiwania obrazu, na ekranie pojawia się ciemność!

Opis ikon.

Na ekranie, który ukazuje się po wybraniu scenariusza, znajduje się u góry z lewej strony mapa terenu, o który będzie się toczyć walka. Po prawej stronie widzimy wnętrze naszego skarbca. U dołu ekranu, pod mapą, znajduje się rząd pięciu ikon. I tak:

Pierwsza od lewej, przedstawiająca dyskietkę, służy do przjścia do ekranu opcji dyskowych. Na tym ekranie pojawiają się trzy możliwości:
- SAVE - zapis stanu gry;
- LOAD - odczyt zapisanego stanu gry lub wczytanie scenariusza;
- drzwi symbolizujące powrót do gry.

Druga ikona od lewej strony umożliwia przegląd stanu posiadania (miast i wojsk).

Wciśnięcie następnej, oznaczonej rombem powoduje pojawienie się linii komunikacyjnych pomiędzy Twoimi miastami.

Czwarta ikona pozwala na powiększenie wybranego fragmentu mapy, zaś ostatnia (dwie strzałki) umożliwia ustawienie poziomu opodatkowania ludności.

Gdy klikniemy ikonę powiększającą fragment mapy, przechodzimy do ekranu, na którym w lewym górnym rogu znajduje się mapka całego terenu gry, pod nią pięć ikon, których znaczenie omówimy za chwilę oraz trójwymiarowy wycinek tejże mapki. Wycinek ten możemy przesuwać za pomocą kliknięć na gadżety na jego obrzeżach.

Kliknięcie ikony przedstawiającej drzwi powoduje powrót do uprzednio opisywanego ekranu głównego.

Wybranie popiersia wojownika pokazuje kolejno oddziały wojska, zaś kuli powiększenie obrazu. Dzięki tej ostatniej opcji można stwierdzić, jak uzbrojone są armie wroga i podjąć właściwą decyzję: bitwa, czy strategiczny odwrót na z góry upatrzone pozycje.

Kliknięcie ikony z dwiema strzałkami powoduje wejście w tryb przeglądu armii i miast.

Magiczna kula (pierwsza ikona z prawej strony, podwójnej wielkości) sygnalizuje pojawianie się komunikatów. Jeśli ujrzymy w niej skrzyżowane miecze, to klikając przełączamy się na ekran bitwy.

Na tej planszy znajduje się pięć ikon które oznaczają kolejno począwszy od lewej strony:

- Atak lub ucieczka z pola bitwy;
- Formowanie szyku bojowego;
- Obrót formacji w lewo;
- Obrót formacji w prawo;
- Strzał.

W celu rozpoczęcia bitwy należy kliknąć ikonę ataku, a następnie chorągiewkę oddziału, który chcemy pchnąć do walki. Kolejnym krokiem jest wskazanie punktu docelowego. Strzelamy klikając chorągiewkę oddziału własnego, potem ikonę strzału, i wreszcie oddział wroga, któremu chcemy dołożyć.

Po prawej stronie ekranu znajdują się symbole przedstawiające charakterystykę zaktywizowanej formacji. Miecze oznaczają siłę oddziału, tarcze, jego opancerzenie, zaś strzały, ilość pocisków do broni miotających. Walka toczy się tak długo, aż któraś ze stron rozpocznie odwrót.



Aby toczyć bitwy, należy najpierw sformować jakieś siły zbrojne. Opisując poszczególne ekrany, z rozpędu pominęliśmy sposób tworzenia armii, jednak co się odwlecze to nie uciecze.

Oddziały wojska można tworzyć jedynie w miastach. W tym celu trzeba klinąć wieżę symbolizującą miasto, co spowoduje przjście do ekranu miasta. Tym ekranem zajmiemy się bliżej w następnej części opisu. Na razie musimy tylko wiedzieć, że można na nim wybrać dwie opcje dotyczące formowania sił zbrojnych. Są to:

- Recruit Army;
- Train Cavalry.

Obie powodują przejście do tego samego ekranu zbrojeń, na którym możemy wybrać rodzaj broni defensywnej i ofensywnej formowanego oddziału. Koszt uzbrajania jest różny w zależności od rodzaju i jakości wojska. Wybranie pierwszej opcji powoduje stworzenie formacji piechoty, zaś drugiej kawalerii. Jeżeli oddział stanowi garnizon jakiegoś miasta, przedstawiony jest na mapie w postaci chorągiewki. Formacja obozująca w polu symbolizowana jest namiotem, zaś oblegająca miasto przeciwnika drewnianą wieżą. (Tu uwaga do p. Tomasza Szczepłka, którego opracowanie służyło mi pomocą przy tworzeniu niniejszego opisu. Słowa "wieża" i "każda" pisze się tak właśnie, a nie "wierza" i "karzda".)

W miastach portowych, oznaczonych kotwicą, armie mogą być załadowane na okręty i przmieszczać się drogą morską. Symbolizowane są wówczas przez mały okręcik. Jeśli napotkają po drodze flotę wroga, wywiązuje się bitwa, na której wynik nie mamy jednak bezpośredniego wpływu.

Opcje dostępne w miastach.

Kliknięcie wieży symbolizującej miasto powoduje przejście do ekranu opcji. Pośrodku znajduje się widok miasta, nad nim opis słowny sytuacji zaopatrzeniowej, stanu zdrowia ludności, jej morale i stosunku do władcy. W dolnej części planszy znajduje się kolejno: stan Twojego skarbca, duża strzałka, której kliknięcie powoduje wybranie pokazanej obok opcji, koszt wyboru tej opcji i ikona z dwiema strzałkami umożliwiająca przegląd dostępnych opcji. Do wyboru mamy wszystkie, lub część poniższych możliwości:

- BUY GRAIN - zakup zboża;
- CITIZEN LEVY - pobór podatków;
- INCREASE LAND - uzdatnienie terenu pod zabudowę miasta;
- IMPROVE HEALTH - poprawa stanu zdrowia mieszkańców;
- STONE WALLS - mury obronne z kamienia;
- TIMBER WALLS - palisada z drewna;
- RECRUIT ARMY - pobór do piechoty;
- TRAIN CAVALRY - pobór do kawalerii;

Do dwóch ostatnich opcji, które zostały już wcześniej opisane, pozwalamy sobie dodać drobną wskazówkę: rasy, które nie przejawiają zdolności w posługiwaniu się bronią miotającą, nie należy uzbrajać w takową. Jest to strata pieniędzy, gdyż efektywność strzałów jest minimalna.

Na podstawie "Amiga Joker" i instrukcji wydanej przez Banana Press opracował *Tomasz Kulbacki.*



GOLDRUNNER

Wpisz EASYMODE na tablicy najlepszych wyników. F9 pozwala przeskakiwać poziomy i czyni Cię nieśmiertelnym, a F8 wyłącza tę opcję.

HARD DRIVIN'

Rozpędź się do max. szybkości i wciśnij N. Od tej chwili masz ułatwione sterowanie, otrzymujesz nieśmiertelność i nieskończony czas.

HAWKEYE

Wciśnij klawisz DEL, a kiedy już umrzesz, znajdziesz się na następnym poziomie.

INTERPHASE

Zacznij grać i wpisz Fenny (z dużej litery F) i wciśnij lewy przycisk myszy. Posiadasz od tej chwili tajny "oglądacz" przedmiotów.

HELTER SKELTER

Ustaw grę na 2 graczy. Gracz nr 1 musi zdobyć extra życie. Teraz obaj mogą stracić wszystkie pozostałe życia. Za chwilę obaj będą mieli po 99 żyć. Kody do leveli:

11 - SPIN,
21 - FLIP,
31 - BALL.

HYBRIS

Na ekranie najlepszych wyników wpisz COMMANDER i wciśnij FIRE. Po chwili wciśnij F10. Masz nieśmiertelność, a klawiszem F8 przechodzisz do następnego poziomu.

KARATE KID 2

P - następny poziom.

JUMPING JACKSON

Na stronie tytułowej wciśnij prawy przycisk myszy.

LAST NINJA 3

Kody do leveli, które trzeba wpisać do tabeli najlepszych wyników:

1 - SUSS, 2 - IMED,
3 - URTI, 4 - BASD,
5 - NOUS, 6 - REOO.

LEONARDO

Kody do leveli:
10 - EMMENTALER,
20 - APLHORN,
30 - MATTERHORN

LEISURE SUIT LARRY

Alt+X pozwala uniknąć pytań "tylko dla dorosłych".

NINJA WARRIORS

Wciśnij CAPS LOCK i wpisz kombinacje podane niżej (możesz wpisać nawet wszystkie naraz) i znów wciśnij CAPS LOCK.

CHEDDAR - nieskończone życia
MONTY PYTHON - nieprzyjaciele wycofują się
STEVE AUSTIN - klawisz S spowalnia tempo gry

TEST DRIVE 2

Wpisz podczas gry:
AERF - szybkie przyspieszenie i extra hamulce,
GASS - przeskok do końca pozimu ze świetnym czasem i wynikiem,
60UTRAN - zobacz se sam

your cheater SANDEY

COMMODORE AMIGA 3000 UX

AT&T UNIX System V Release 4

TCP/IP / NFS / RFS

X Window System / Open Look





Commodore

KONCERN TECHNOLOGII